AFIT/GE/ENG/97D-22

Applications of Unsupervised Clustering Algorithms

to Aircraft Identification Using High Range Resolution Radar

THESIS
Dzung Tri Pham
Captain, USAF   DTIC QUALITY INSPECTED 2

AFIT/GE/ENG/97D-22

19980130 148

The views expressed in this thesis are those of the author and do not reflect the official policy or

position of the Department of Defense or the U. S. Government.

Applications of Unsupervised Clustering Algorithms

to Aircraft Identification Using High Range Resolution Radar

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Dzung Tri Pham, B.S.E.E., M.B.A.
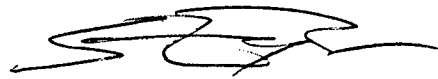
Captain, USAF

December 1997

APPLICATIONS OF UNSUPERVISED CLUSTERING ALGORITHMS

TO AIRCRAFT IDENTIFICATION USING HIGH RANGE RESOLUTION RADAR

Dzung Tri Pham, B.S.E.E., M.B.A.

Captain, USAF

Approved:

_____   6 Nov 97
Dr. Steven K. Rogers   Date
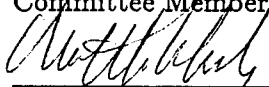Thesis Advisor

_____   6 Nov 97
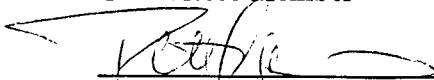Dr. Martin P. DeSimio   Date
Committee Member

_____   6 Nov 97
Dr. Matthew Kabrisky   Date
Committee Member

_____   6 Nov 97
Dr. Peter Collins   Date
Committee Member

Table of Contents

List of Figures

viii

List of Tables

Acknowledgements

Like the saying goes, "All good things must come to an end." This has been an enjoyable and valuable experience and I will never regret the time I spent completing this work. I worry not of the past but what the future will bring. I look forward to the next challenge that lie ahead, but would first like to thank the people that made the challenges and experiences of the past enjoyable.

First, I would like to thank my advisors, Dr. Steven Rogers and Dr. Martin Desimio for creating a wonderful learning environment and for their truely valuable guidance. I would also like to thank my other two committee members, Dr. Mathew Kabriski and Dr. Peter Collins, and my friend Wilburn Baker for their time and effort in bringing this effort to its finished form. With the greatest respect I thank these men for their scholarship, experience and guidance.

Secondly, I would like to thank all my friends that kept me sane through this experience. To my friends, P.J. Barrett, Corike Toxopeus, Warren Nuibe, Don Hill, Sue Ransom, Dave Kaneshiro and Julie Siwik, I thank you for being there during all the rough times and also the good times. There are no words that could ever truly explain what you all mean to me and your friendship and kindness have been immeasurable.

I also wish to thank my Mom, Nu, for encouraging me to believe in my self and my abilities. Her strength and support have made it possible for me to overcome my shortcomings.

Finally, I would like to thank my fellow classmates who had to endure the same challenges that have made AFIT so "enjoyable". Their friendships in this endeavor have made this process alot of fun and as painless as possible. I thank you all!

<div align="right">Dzung Tri Pham</div>

Abstract

Identification of aircraft from high range resolution (HRR) radar range profiles requires a database of information capturing the variability of the individual range profiles as a function of viewing aspect. This database can be a collection of individual signatures or a collection of average signatures distributed over the region of viewing aspect of interest. An efficient database is one which captures the intrinsic variability of the HRR signatures without either excessive redundancy (over-characterization) typical of single-signature databases, or without the loss of information (under-characterization) common when averaging arbitrary groups of signatures.

The identification of "natural" clustering of similar HRR signatures provides a means for creating efficient databases of either individual signatures, or of signature templates. Using a k-means and the Kohonen self organizing feature net, we identify the natural clustering of the HRR radar range profiles into groups of similar signatures based on the match quality metric (Euclidean distance) used within a Vector Quantizer (VQ) classification algorithm. This greatly reduces the redundancy in such databases while retaining classification performance.

Such clusters can be useful in template-based algorithms where groups of signatures are averaged to produce a template. Instead of basing the group of signatures to be averaged on arbitrary regions of viewing aspect, the averages are taken over the signatures contained in the natural clusters which have been identified.

The benefits of applying natural cluster identification to individual-signature HRR data preparation are decreased algorithm memory and computational requirements with a consequent decrease

in the time required to perform identification calculations. When applied to template databases the benefits are improved identification performance.

This paper describes the techniques used for identifying HRR signature clusters, and describes the statistical properties of such clusters.

Applications of Unsupervised Clustering Algorithms

to Aircraft Identification Using High Range Resolution Radar

I. Introduction

1.1 Background

The prevention of fratricide and the assured destruction of enemy assets are dependent on

timely, accurate, and high confidence identification decisions. Proper target identification, there-

fore, is central to the successful execution of all warfighting missions, regardless of operating unit

and operational arena. The information received from identification systems is of critical impor-

tance to the warfighter as he makes key engagement decisions in high track density environments

where the warfighter's ability to sort out friendly, hostile and neutral forces is severely stressed.

The objective of combat identification is twofold: to maximize the effectiveness of our weapons

systems against hostile tracks while at the same time preventing fratricide. Combat identification

is not a "system"; it is a capability. No single system can provide one-hundred percent positive

ID. Combat identification is a combination of doctrine, tactics, techniques and procedures; sen-

sors; multi-sensor integration; reliable communications; and identification processors that provide

situational awareness with a high degree of confidence.

One promising technology that is currently of great interest to the Air Force is non-cooperative

target identification (NCTI) using high range resolution (HRR) radar. NCTI are ID systems that

require no response from the target in order to make an identification, unlike cooperative ID systems,

classified as Question and Answer systems, that require the target to respond with a correct answer

when cued. Cooperative ID systems require the costly procurement and vulnerability that comes with the requirement of using transmitters or transponders. These transmitters or transponders are vulnerable to failure due to mechanical means, electrical means or even due to incorrect code response caused by human error. A typical cooperative system is the Identification Friend or Foe (IFF) system.

An NCTI system requires no such transmitters or transponders, and is free of such possible failures. Many possible sensor systems may be used in developing an NCTI system. They range from electro-optical identification systems that operate in the visual or infrared spectrum to radar systems such as Jet Engine Modulation (JEM) that use the modulated radar return pattern due to the engines as a recognition technique. One particular system that has been identified as having great potential for use in NCTI is the HRR radar.

The HRR Mode uses a high resolution profile generated by the radar to correlate with target profile data stored in a library. These profiles provide a long range, near all aspect capability for radar based identification. Unlike conventional radar, which is used to estimate the relative range, heading, and velocity of a target, HRR radar is specifically designed to provide an electro-magnetic profile of the target along the radar-target vector or direction of propagation (32). A pulse-compression radar can provide sufficient range-resolution to obtain an electrical profile of the target, as determined by the target's major scattering centers. One should not expect the radar to provide the same target details as seen visually, but what is seen electromagnetically. Conceptually, an HRR range profile is obtained by dividing the range dimension (along the direction of propagation) into range bins of some width, and measuring the backscatter energy from all parts

of the target that are within that range bin. As the resolution of the radar increases, the width of each range bin decreases. An example of a range profile is given in Figure 1.
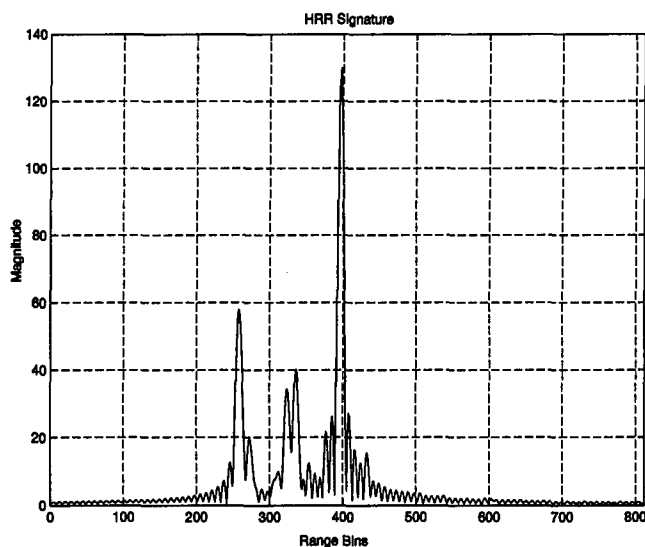
Figure 1. HRR Range Profile

HRR systems are able to relieve some of the drawbacks that are evident with current JEM systems. HRR offers long range, all aspect identification, it does not require engine returns at the target frontal or rear aspect, and it also provides improved accuracy. But, the main drawback comes with the development of the look up library for the system. The library must either be built through expensive and covert data collections of each possible target at all possible aspects or it must be generated using computational electromagnetic techniques and codes. This means a very large and expensive data base for a problem of any significant size. Either way we choose to gather the profiles, the task is difficult and not an easy barrier to overcome.

The approaches that are most commonly followed to meet the ID requirement generally fall into one of three broad categories. The three main areas of emphasis for researchers and developers

are usually placed on 1) specific mathematical methods or algorithms, 2) developing data bases (for template matching algorithms), or 3) developing radar systems or sensors (9). One such group of developers and researchers is the Wright Laboratory Combat Identification Branch, WL/AAZI. They in concert with Hughes Aircraft Company, are currently pursuing an HRR system for aircraft identification. They are developing a new synthetic range profile database and a template based Adaptive Gaussian Classifier (AGC) algorithm that are showing much promise (1).

1.2 Problem Statement

This thesis investigates the use of clustering techniques to enhance algorithm performance with synthetic database range profiles and to reduce the total number of required templates per class that must be stored in the database. By evaluating the "natural" signature clusters in the synthetic database we will eliminate redundancy between individual signatures. We will also increase the accuracy in constructing signature templates by grouping like signatures during template construction. This promises improved performance in template matching algorithms, such as the AGC. The AGC uses templates created by using 5 degrees azimuth by 5 degrees elevation windows and finding the mean signature in the window for use as a template to represent all signatures that occur in the window.

1.3 Summary of Current Knowledge

Non-Cooperative Target Identification has received a great deal of attention in the past years, and as a result, considerable work has been accomplished in this area. Rather than taking a broad overview of the entire field, we will look at some specific work accomplished in the recent years with the HRR radar technology. Most researchers in this area work on developing new or better

automatic target recognition algorithms. Others work on improving the associated sensors. We will focus on creating a better database. We will work on improving algorithm performance and the quality of the database through database reduction via the clustering of the range profiles. This author is aware of only two efforts in the past few years in this area. The first of these efforts was in 1995, by Frederick Beckner, et al. where they employed an algorithm using a "Cluster Seed" to create match lists of all the profiles using a similarity measure (2). The second of these effort was in 1997, by Rick Venger, where he applied a Kohonen Net to obtain clusters of the range profiles (35).

To accomplish the tasks of improving classification performance, reduction of the database redundancy, and improving the quality of the representation required in the database, this thesis will draw on the before mentioned experimental results and also from a range of theoretical and experimental results developed in the area of clustering and dimensionality reduction. The summary of the techniques and results from the two above experiments will be expanded in the following sections and the key theoretical components will be summarized in these sections and expanded in greater detail in Chapter II.

1.3.1 Clustering. Clustering analysis is one of the basic tools for identifying structure in data. Clustering tries to partition objects in N disjointed subsets using some sort of similarity metric. That is, if we had M objects, where $M > N$, we want to partition this set of M objects into N subsets, where members of each N subsets are said to belong to the same cluster and are more similar to each other than to members of different subsets or clusters. The objective is to automatically search a database containing a large number of different objects and to group these objects into subsets of similar objects called clusters.

5

The ability to form such clusters has a number of applications in target recognition. Rather than just simply using the whole data set or simply choosing an arbitrary subset of the data points as a basis set, one can use clustering techniques to find a basis set which more accurately represents the distribution of the data points. We can find these basis sets, by finding the cluster centers which are the mean data points that can best represent all members of a single cluster. By finding the mean data point of each cluster, we can create templates that can reduce the amount of data that need to be stored with minimum loss of information. These templates are then used in a template matching algorithm such as the AGC.

1.3.2 Clustering using Cluster Seed. One technique for clustering HRR radar signatures is the "Cluster Seed" method. This method was developed by Beckner, et al. in 1995 with good results. This technique is described in fuller detail in Chapter II. This signature clustering technique was then used to analyze signature clusters in a number of sectors in the database. The database they used in this work is a subset of the Automatic Radar Target Identification (ARTI) Phase 3 database of measured ground-to-air HRR radar signatures. This subset of the ARTI database contains over 920,000 individual radar signatures from 12 different fighter-sized aircraft.

Some of their findings from the analysis are as follows. Replacement of the cluster signatures by the cluster center was shown to reduce the database size by a factor of between 2 and 3 in most cases examined. They also postulated that the signature clustering technique will prove valuable in the formation of mean signature templates used by some identification algorithms by providing a means of selecting the number and location of the templates they will improve algorithm performance and not degrade performance by averaging nonsimilar signatures as can occur when forming templates from ad-hoc regions in aspect space (2).

1.3.3 Clustering using Kohonen Net. An alternate, more common method for clustering HRR radar signature is the Kohonen Net, also called a self-organizing feature map. A Kohonen Net is an unsupervised algorithm that can process unlabeled data, i.e., data where the desired classification is unknown. It can map high dimensional data onto a two-dimensional output or cluster layer. When the algorithm is properly trained, cluster centers corresponding to nearby points on the feature map grid have nearby locations in input space. Further description of the Kohonen Net is in Chapter II.

In July 1997, Venger investigated the presence of a "natural" synthetic generated HRR data clustering. His objective was to determine the potential of enhanced algorithm performance with a synthetic database and to determine the potential of reduction in required templates needed per target class. The data used for training in this study was a 6 class set from the December 1996 synthetic signatures database release from Wright Laboratories. The aspect window investigated was a window of azimuth from 0 degree to 25 degrees and elevation from 0 degree to -20 degrees.

To obtain clusters, Venger, used a standard Kohonen Net with 5X4 output nodes to obtain the same number of templates as the baseline case that used 20 templates based on 5X5 degrees signatures averaging windows. Once clusters and templates were determined he assessed the AGC algorithm performance on the clustered templates database relative to the standard 5X5 degrees subsectoring approach. The test data was ARTI III/Phase 4/T1 over the same aspect region as used in training.

The following is a summary of results from the study by Venger. The results showed the algorithm performance on templates created by clustering synthetic signatures was not enhanced, but comparable to the baseline using the 5X5 degrees subsectoring approach. Venger showed that

the average probability of declaration, Pd, and the average probability of confidence, Pc, obtained with clustering was within ±1% of the baseline. The second observation reported by Venger was that there were opportunities for reduction in number of templates through the use of clusters, but a reduction percentage was not given. The results showed that the clusters for some targets were diffuse and overlapping, covering a large areas of aspects region. These clusters could possibly be combined into one, therefore expanding the aspect coverage per template. For example, rather than using 5X5 degrees templates, a larger degree window templates might suffice (35).

1.4 Scope

Computer-synthesized range profiles of three aircraft, generated at 1 degree increments in a 30X15 degrees aspect angle window of interest (a total of 1116 signature profiles per aircraft) will be used as training data to create templates using several clustering techniques. These templates will then be used in template matching algorithms and tested using ARTI III measured data on the same three classes of aircraft (over 15,000 signatures total). By the use of clustering techniques to create templates, this thesis analyzes the ability to improve classification performance and the ability to reduce the number of templates required to maintain baseline performance. While the clustering techniques used are applied to this particular problem, the techniques are applicable to other problems where the grouping of "similar" objects might help with classification or reduce redundancy to minimize the amount of data stored.

1.5 Approach

The approach taken in this investigation is composed of two steps. The first step is to implement a batch K-means algorithm and the Kohonen clustering algorithm described by Rogers,

et al. (29) and validate them using some simple known toy problems before applying them to the HRR problem. The second step is to apply these techniques to the HRR problem to determine if classification performance can be increased and to determine if the amount of data stored can be reduced while maintaining equivalent performance levels.

1.6 Research Objectives

There are five research objectives for this thesis:

1. Implement the batch K-means algorithm outlined by Rogers, et al. (29) and verify its performance on known problems.

2. Implement the Kohonen clustering algorithm described by Rogers, et al. (29) and verify its performance on known problems.

3. Apply the two techniques to the HRR problem to generate templates for use in a classification algorithm.

4. Develop a sufficient understanding of how each method works to be able to draw appropriate inferences from the classification results, beyond the knowledge of either improving and lowering performance.

5. Determine if reduction in the data stored can be accomplished without impacting levels of performance.

1.7 Thesis Organization

Chapter II begins with a description of the statistical and mathematical theories which support the methodology and results of this thesis. Chapter III describes the actual implementation

9

of these theories. Chapter IV presents procedures used to verify the techniques described in Chapter III and discusses the results of applying these techniques to known toy problems. Chapter V presents and discusses the the results obtained by applying these techniques to the HRR problems. Finally, Chapter VI summarizes the results obtained in this investigation and suggests future work that could be done.

1.8 Summary

The need for positive, timely, and reliable identification of friends, hostile and neutral aircraft is of vital importance to today's warfighters. The techniques investigated in this thesis will help to improve classification accuracy while reducing the amount of data required to do this classification. The techniques used for this problem should be applicable not only to the HRR problems, but to many other pattern recognition problems.

II. Theory

2.1 Introduction

Before we proceed further, some basic understanding of the techniques used and discussed must be reviewed. In the following chapter we will develope the theoretical concepts necessary to understand the methods used to accomplish our objectives. In the sections to follow, these areas are discussed:

- Basic Pattern Recognition Theory

- Bayes Theory

- Discriminant Function and Bayes Error

- Gaussian Classifier

- Density Estimation and Bayes Error Estimation

- Confidence Intervals

- The two paradigms: Supervised and Unsupervised

- Clustering

- Similarity Measures

- Clustering, Dimensionality, and Data Reduction

- The Batch k-means Algorithm

- The Kohonen Self Organizing Feature Maps

- Clustering Seed

2.2 Pattern Recognition

What is Pattern Recognition? Fukunaga states that the goal of pattern recognition is to clarify the complicated mechanisms of the decision-making processes of a human being and to automate these functions using computers (13). Duda and Hart describe this task as "machine perception", or the ability of machines to perceive their environment where many of these "machine perception" cases, involve pattern classification or the assignment of a physical object or event to one of several prespecified categories (8). No matter how you describe Pattern Recognition, the task of coming up with a solution using a computer is in many cases immensely difficult. The apparent ease with which humans perform the perceptual task is encouraging, but modeling this seemingly simple task using a computer is where the difficulties come in.

To come up with a possible solution to the pattern recognition problem one must come up with a general framework to formulate our solutions. Often the information given to solve the problem and the decision procedure used to develop a solution are statistical in nature. The most general, and most natural framework in which to formulate solutions to pattern recognition problems is a statistical one, which recognizes the probabilistic nature both of the information we seek to process, and of the form in which we should express the results (3)(13)(8)(34). Therefore we will base our general understanding of the field of pattern recognition on the framework of the study of general statistical-based decision methods. In the remainder of this section we will introduce some of the basic concepts of the statistical approach to pattern recognition, in preparation for later chapters.

Bayes decision theory is a fundamental statistical approach to the problem of pattern recognition. The basic assumptions that we make are that the problem is posed in probabilistic terms and that all of the relevant probability values are known. We will base our treatment of Bayes

12

Theory on a similar outline as that found in Duda and Hart, to which readers are referred for a more detailed account of these topics (8).

2.2.1 Bayes Theory. Let us consider the problem of designing a classifier to determine two kinds of aircraft, a F-15 and a Mig-29. Suppose a pilot flying an intercept mission comes upon an unknown aircraft and needs to determine if it's a friendly F-15 or a hostile Mig-29. Using decision theory terminology, we say that as we encounter an unknown aircraft that the nature is in one of two possible states: either the aircraft is a F-15 or a Mig-29. We let ω_i denote the state of nature, with $\omega_i = \omega_1$ for the F-15 and $\omega_i = \omega_2$ for the Mig-29. Because the state of nature is so unpredictable, we consider ω_i to be a random variable.

If the chance of encountering a F-15 and a Mig-29 were known, we would say that there is some *a priori* probability $P(\omega_1)$ that the unknown aircraft is a F-15, and there is some *a priori* probability $P(\omega_2)$ that the unknown aircraft is a Mig-29. These *a priori* probabilities reflect our prior knowledge of how likely we are to encounter a F-15 or a Mig-29 before we actually find out what it is. We also know that $P(\omega_1)$ and $P(\omega_2)$ are nonnegative and must sum to one.

Suppose now, we are required to decide what the unknown aircraft is. The only information we have to aid us in our decision is the *a priori* probabilities. Then it is reasonable to use the following decision rule given the little amount of information we are allowed: The decision would be ω_1 if $P(\omega_1) > P(\omega_2)$; otherwise we would choose ω_2.

So we would always choose the state that has a larger *a priori* probability, even though we know that there are two possible types of aircraft and that both aircraft will be encountered eventually. How well this decision rule works depends on the value of the *a priori* probabilities. If $P(\omega_1)$ is much larger than $P(\omega_2)$ then we will be right a majority of the time. If the *a priori*

13

probabilities are equal than we will only be right fifty percent of the time and the probability of making an error is fifty percent. In general, the probability of error is the smaller of the two *a priori* probabilities, and we shall see later in the section 2.2 that under these conditions no other decision rule can achieve a smaller probability of error.

Fortunately, one does not usually have to make a decision with such a little amount of information. In our example, we can use the radar of our aircraft to get an electromagnetic scattering x from our unknown target as additional information or features. Different aircraft encountered will yield different scattering returns, and it is natural to express this in probabilistic terms; we consider x to be a continuous random variable whose distribution depends on the state of nature. Let $p(x|\omega_j)$ be the state-conditional probability density function for x, the probability density function for x given that the state of nature is ω_j. The difference between $p(x|\omega_1)$ and $p(x|\omega_2)$ describes the difference in scattering return between the F-15 and the Mig-29.

Given the measurement x and that we know both the *a priori* probabilities $P(\omega_j)$ and the conditional densities $p(x|\omega_j)$, we are able to determine the probability that the measured aircraft is a member of class (ω_j). This probability, which is $P(\omega_j|x)$, is determined by Bayes Rule:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)}. \tag{1}$$

where

$$p(x) = \sum_{i=1}^{2} p(x|\omega_i)P(\omega_i). \tag{2}$$

Bayes rule shows that by knowing the value of x, we can change the *a priori* probability $P(\omega_j)$ to the *a posteriori* probability $P(\omega_j|x)$. The *a posteriori* probability is a quantity of great interest since it allows us to make optimal decisions regarding the class membership of any new measurements.

In particular, in the next section we will discuss how deciding the class membership by using the largest *a posteriori* probability will minimize the probability of misclassification of that object.

The scalar x can also be more than one feature and can be replaced by the feature vector $\mathbf{x}$. Then the *a posteriori* probability $P(\omega_j|\mathbf{x})$ can be computed from $p(\mathbf{x}|\omega_j)$ by Bayes rule:

$$P(\omega_i|\mathbf{x}) = \frac{\mathbf{p(x|\omega_i)P(\omega_i)}}{\mathbf{p(x)}}. \tag{3}$$

where

$$p(\mathbf{x}) = \sum_{i=1}^{2} \mathbf{p(x|\omega_i)P(\omega_i)}. \tag{4}$$

There are many different ways we can choose to represent our pattern classifiers. So far, our focus has been on the relative sizes of the *a posteriori* probability and this has determined the class membership. This observation gives us an opportunity to reformulate our classification process in terms of discriminant functions $g_i(x)$, where $i = 1, ..., c$ and c is the number of classes. Our Bayes classifier is now easily represented as $g_i(x) = P(\omega_i|x)$. Now our decision rule is said to assign a feature vector x to class (ω_i) if

$$g_i(x) > g_j(x) \quad \text{for all } j \neq i. \tag{5}$$

The classifier is now viewed as a machine that computes c discriminant functions and selects the category corresponding to the largest discriminant. No matter what we use as a decision rule, the end effect is to divide the feature space into c decision regions, $R_1, ..., R_c$. Now we can say that if $g_i(x) > g_j(x)$, for all $j \neq i$, then x is in R_i and a member of class ω_i. The boundaries that separate the regions are called decision boundaries and are surfaces in feature space where the largest discriminant functions are equal, $g_i(x) = g_j(x)$.

2.2.2 Discriminant Function and Bayes Error. Let's reexamine our example of deciding whether an unknown aircraft is a F-15 or a Mig-29. If we get a measurement x for which $P(\omega_1|x)$ is greater than that of $P(\omega_2|x)$, we would declare our unknown aircraft to be a member of class ω_1. But, by making this choice of membership, we know there is a possibility of an error occurring that is equal to $P(\omega_2|x)$. So whenever we observe a particular value of x, and make our decision the following is the inherent error of our decision:

$$P(\text{error}|x) = \begin{cases} P(\omega_1|x) & \text{if we decide } \omega_2 \\ P(\omega_2|x) & \text{if we decide } \omega_1. \end{cases} \qquad (6)$$

Clearly this will minimize the probability of error for a value of x, but will this decision rule minimize the average probability of error? Yes, the average probability of error is given by

$$\begin{aligned} P(error) &= \int_{-\infty}^{\infty} P(error, x)\, dx \\ &= \int_{-\infty}^{\infty} P(error|x) p(x)\, dx \end{aligned} \qquad (7)$$

and if we minimize $P(error|x)$ for all values of x, than the integral will be a minimal also. This justifies the Bayes decision rule and will result in the minimum probability of error.

By thinking of a classifier as a device for partitioning feature space into decision regions, we can obtain additional insight into the operation of a Bayes classifier. Go back to the two-class problem, and suppose our classifier has divided the feature space into two regions, R_1 and R_2. This is illustrated in Figure 2. There are two ways a classification error can occur; either the measurement x falls in R_1 when the true state is ω_2 or when x falls in R_2 and the true state is ω_1

since these events are mutually exclusive and exhaustive,

$$
\begin{aligned}
P(error) \quad &= P(x \in R_2, \omega_1) + P(x \in R_1, \omega_2) \\
&= P(x \in R_2, \omega_1)P(\omega_1) + P(x \in R_1, \omega_2)P(\omega_2) \qquad (8) \\
&= \int_{\mathcal{R}_2} p(x|\omega_1)P(\omega_1)\,dx + \int_{\mathcal{R}_1} p(x|\omega_2)P(\omega_2)\,dx
\end{aligned}
$$

where $P(x \in R_1, \omega_2)$ is the joint probability of x being assigned to class ω_1 and the true class being ω_2. Thus, if $p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2)$ for a given x, we should choose the regions R_1 and R_2 such that x is in R_1 as this gives a smaller contribution to the error. We recognize this as the decision rule given by Equation (5) for minimizing the probability of misclassification. The same result can be seen graphically in Figure 2, in which misclassification errors arise from the shaded region. By choosing the decision boundary to coincide with the value of x at which the two distributions cross we minimize the area corresponding to classifying each new pattern x using Equation (5), which is equivalent to assigning each pattern to the class having the largest posterior probability.
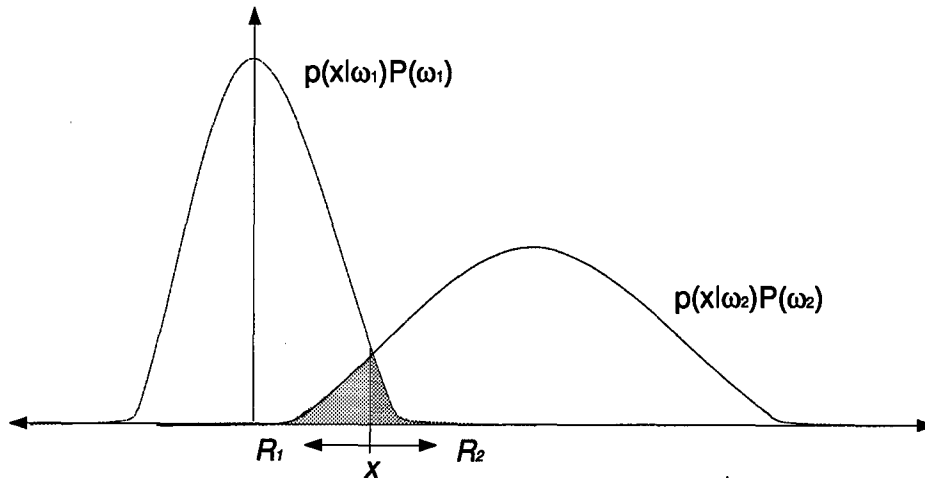
Figure 2. Components of the probability of error

2.2.3 Gaussian Classifier. The theory presented in the previous sections provides the basis for developing a specific form of classifier. The Gaussian or Quadratic classifier is so named because of the mathematical form of its decision boundaries. This classifier is parametric, in that it is specified in terms of the parameters (mean and covariance) of the class distributions.

The Gaussian classifier is best described as a template-based pattern recognizer and consists of two main parts. The first part is the targets template library. The target library contains statistical information on each target, the mean μ and the covariance Σ of each aircraft for each range bin. The target library consists of many different aircraft at many different aspects.

The second part of the Gaussian classifier includes the discriminant function and the decision analysis. As an unknown target is observed, it is compared against the templates in the target library. Everytime the unknown range profile is compared against a database template, a discriminant metric is formed quantifying the similarities between the unknown range profile and the template.

Using the simple two class problem as described in Section 2.2.1, we wish to identify between two possible unknown aircraft, the F-15 and the Mig-29. With the Gaussian classifier, the quantity used in hypothesis testing is referred to as the minus-log likelihood ratio (13)(8). The likelihood ratio is written in terms of the input measurement to be classified, x, the mean of class one μ_1, the mean of class two μ_2, and their associated covariances Σ_1 and Σ_2 and is given as:

$$g(x) = \frac{1}{2}(x - \mu_1)^t \Sigma_1^{-1}(x - \mu 1) - \frac{1}{2}(x - \mu_2)^t \Sigma_2^{-1}(x - \mu_2) + \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|} \qquad (9)$$

Assuming both class 1 and class 2 are equally likely, the decision rule would be if $g(x) > 0$ then we declare the unknown target as class 1, else we declare it class 2.

18

Now consider the multi-class problem, where the likelihood ratio can be written as:

$$g_n(x) = \frac{1}{2}(x - \mu_n)^t \Sigma_n^{-1}(x - \mu_n) - \frac{1}{2}\ln|\Sigma_n| - \ln P_n \qquad (10)$$

where P_n is the *a priori* probability for each of the classes. Using this discriminant function, the unknown signature is compared against all of the templates. After all comparisons are made, the unknown signature is declared to belong to the class which results in the minimum discriminant value.

2.2.4 Density Estimation and Bayes Error Estimation.   Even though the best possible classifier to achieve Bayes error uses the *a posteriori* probabilities as the discriminant function, it is very unlikely that we know the class-conditional probability distribution functions (pdf) for all the classes. We must try to make an estimate of these parameters, and there are two possible approaches we can use to accomplish this. The first approach involves parametric methods in which a specific functional form for the pdf is assumed. Then data samples are used to update the initial model to optimize the pdf estimate. The drawback to this approach is that the assumption of what initial function form for the pdf maybe incapable of providing a good representation of the true pdf (8)(13)(3)(34).

The second approach does not have this drawback. It involves a non-parametric estimation that does not assume any particular functional form for the pdf. The functional form for the pdf is determined entirely by the data sample. The two most common non-parametric approaches are the k-nearest neighbor (k-NN) and the Parzen window estimators. The drawback to non-parametric approaches is that the estimate is less reliable if the data has a large bias or variance since a small

number of neighboring samples are used. Therefore the non-parametric methods may require a large amount of data to achieve a good estimate of the pdf (8)(13)(3)(34).

So far we have considered parametric and non-parametric techniques for estimating the Bayesian descriminant function. Now we want to address the general problem of estimating the error rate for a classifier in a pattern recognition problem when only a finite set of data is available. We want to accurately predict the average probability of error when only a finite amount of data is available so that when future data samples are classified we will achieve the same error rate.

Martin, following on earlier work done by Fukunaga (13), used Resubstitution (R) and leave-one-out (L) estimates to produce upper and lower bounds on the Bayes error rate respectively. Resubstitution is a method that uses the complete set of available data to train and test, while the leave-one-out method trains with all but one available sample which is used to test and then this process is repeated until all available samples are used as the single test sample. Then using a Bayes decision rule and applying two non-parametric methods, Parzen and k-nearest neighbor R and L pdf estimates, he estimates the error rate. A rather thorough discussion of this topic is given in Martin's thesis (26).

2.2.5 Confidence Intervals. Unfortunately bounding the Bayes error rate can be analytically intractable in high dimensional problems, problems with more then two classes, or problems where the training set and data set are different (synthetic training set and measured test set). To overcome this, Fukunaga suggests using a Monte Carlo analysis to come up with an estimate of the predicted error rate for the classifier (13). Since the performance of the classifier with the test set is an estimate itself and is a random variable with some variance associated with it, this uncertainty may be quantified with the use of confidence intervals.

The basic procedure for producing the required confidence intervals is given by Papoulis (28). When applying a confidence interval to the performance of the classifier, one is estimating the variance of a proportion, where the proportion is represented by the test set. The performance of the proportion is the expected value of the correct classification of the classifier if applied to new data. The proportion behaves as a Bernoulli random variable with a binary output: either correct or incorrect classification (10).

The procedure uses the result of one Monte Carlo trial as an estimate of the mean of a Bernoulli process. The more trials we run, the more confidence we have in our estimate. This means that as we run more trials, the confidence interval should converge and we should close in on the actual value of the mean.

To produce the confidence interval the following equations are used:

$$\tilde{\sigma}^2 = \frac{\tilde{p}(1-\tilde{p})}{n}, \tag{11}$$

$$p \approx \tilde{p} \pm z_u \sqrt{\frac{\tilde{p}(1-\tilde{p})}{n}}, \tag{12}$$

where $\tilde{\sigma}^2$ is the estimated variance, $\tilde{p}$ equals the average classification rate for n trials, and z_u is the test statistic. For a 97.5% confidence interval the value of z_u is given as 1.96.

2.2.6   The Two Paradigms.

2.2.6.1   Supervised Pattern Recognition.   In supervised pattern recognition there exists some set of patterns, and the individual classes of which are already known. The labeled data is then divided into two sets. One of the set, known as the training set, is used to derive

the classification algorithm. The other set, known as the test set, is used to test the classification algorithm. Since the test set is composed of labeled data, one can evaluate the performance of the algorithm. In pattern recognition problems of this sort, one often thinks of the results as being evaluated by a "teacher" whose output dictates suitable modifications to the algorithms. Once an acceptable performance level for the algorithm is achieved, the algorithm can be used on unlabeled data. This procedure is illustrated in Figure 3 (34).
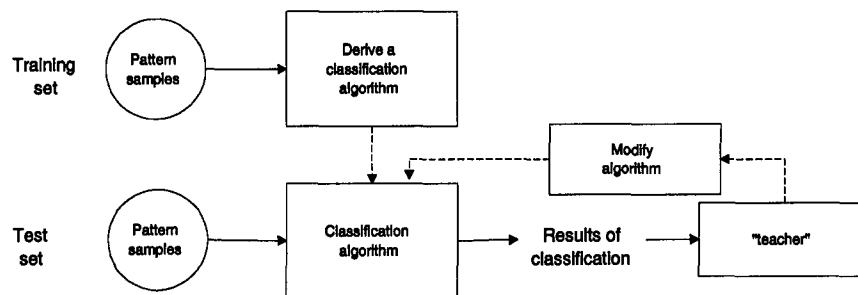
Figure 3. Flow chart of the design procedure in supervised pattern recognition

2.2.6.2 Unsupervised Pattern Recognition.   Unsupervised pattern recognition or clustering is a problem in which the classes of all of the available patterns are unknown, and perhaps even the number of the classes is unknown. In the unsupervised pattern recognition problem, one attempts to find classes of patterns with similar properties where sometimes even these properties may be unknown. Therefore, there are no labeled data available for design of our classifier or to test it. The only hope is that data from the same class are somehow self-similar, but different between classes. Needless to say, the unsupervised problem is a much more difficult one than that of the supervised. Nevertheless, useful algorithms have been developed in the area and success depends, to a large extent, on the ability to learn the structure or pattern in the data in high-dimensional spaces (34).

22

2.3 Clustering

In the preceding sections, we have discussed the basic design theory for pattern recognition. The theories behind designing a Bayesian classifier, parameter and density estimations and Bayesian error have been discussed. Throughout these discussions our assumption has been that there exists a training set of labeled samples. In this section, we will focus our attention on the classification of samples without the aid of such a labeled training set. But the problem is not only to classify the given data, but also to define the classes or the number of classes. We will refer to this kind of classification as clustering or unsupervised classification.

Clustering algorithms attempt to find the underlying relationship of input patterns using the assumption that inputs of the same class will naturally cluster together. How effectively the algorithm clusters depends greatly on the similarity measures used. Ideally, when the algorithm is effective, we achieve an optimal number of clusters that have samples that are similar to each other based on our similarity metric, with small variance. Each cluster is represented by a single point that is called a cluster center or codeword. A full set of codewords is known as a codebook. This codebook, once created, can be used to classify new samples by comparing the input sample to each codeword. The codeword which is the most similar to the input by use of the similarity metric is the class to which the input belongs (29).

In the following sections, two well known unsupervised learning algorithms and one relatively unknown but unique clustering algorithm will be introduced. The three algorithms are k-means, Kohonen's self organizing maps, and the Clustering Seed algorithm.

2.3.1 Similarity Measures.

We stated that the affectiveness of our clustering algorithm depends on the similarity measures we use. What we mean by this is in what sense are we to say that

the samples in one cluster are more alike with each other, than like samples in the other clusters. The similarity metric must be chosen so that it is a good discriminator between our different classes and inherent to our particular problem or classifier (13). The issue is how to measure the similarity between the samples. The distance between them is probably the most obvious measure of the similarity between two samples.

Euclidean distance has been suggested as the most commonly used similarity metric (25). If we use euclidean distance as our metric to determine similarity between our samples, what are some of the pitfalls with choosing such a similarity metric? If the euclidean distance between two samples is less then some threshold t, then the two samples are said to belong to the same cluster. It is obvious that the value of the threshold t is very important in determining if the clustering algorithm is affective. If t is too large than all the samples will be assigned to just one cluster. If the value of t is to small than each sample will form its own isolated cluster. To obtain the optimal amount of clusters, we have to carefully choose the value of t that will obtain the desired "natural" clusters.

A less obvious pitfall, is that by choosing euclidean distance as our metric, we have made some important assumptions. The choice of euclidean distance implies that the feature space is isotropic. Also by choosing euclidean distance, our clusters are assumed to be invariant to translations or rotations, but will not be invariant to linear transformation in general, or to other transformations that distort the distance relationships. So, for clusters to be meaningful, they need to be invariant to transformations natural to the problem. These are just some of the possible pitfalls that we must keep in mind when designing our clustering algorithms (8).

For our investigation we propose two possible discriminant functions. We can either use a quadratic discriminant function where we take into account the variances of the data or we could use a euclidean discriminant function where all the variances are set equal to one. Kosir and Dewall in October of 1997 investigated this issue using a six class HRR problem trained on synthetic signatures and tested on measured signatures using the Adaptive Gaussian Classifier (19). Their results are shown in Figure 4. They concluded that the quadratic discriminant did worse in all aspects than euclidean discriminant function. The use of variance in the discriminant actually caused the classification performance to be lower. They concluded that since the training set was synthetic, the variance of the training set did not properly represent the variance of the measured test set and using the variance did not add information but instead added noise. Therefore, we will use euclidean distance for our investigations.
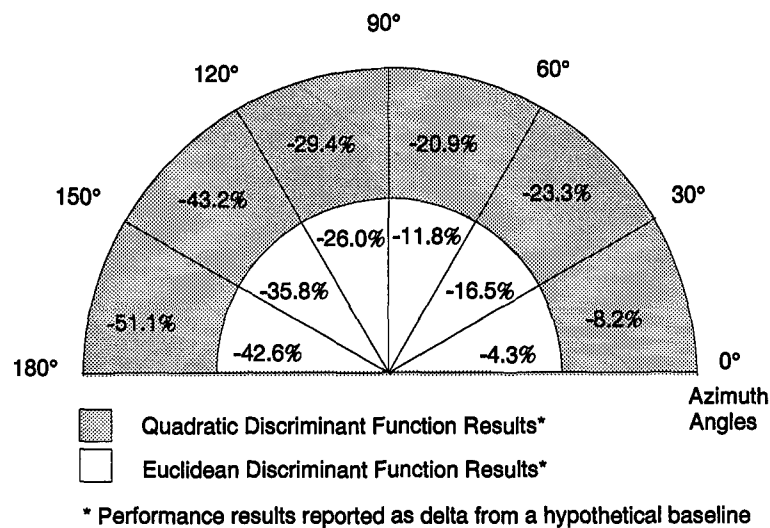
Figure 4. Average performance results comparison of Quadratic versus Euclidean discriminant functions for a six class problem

2.3.2 Clustering, Dimensionality, and Data Reduction. One of the biggest problems encountered in pattern recognition is the curse of dimensionality. As the dimensionality of data

increases, the number of data samples required to properly train an algorithm increases and the combination of features to be examined increases exponentially, resulting in computational cost (12)(4)(11)(16)(15). A variety of methods for dimensionality reduction have been proposed (21)(14)(33)(31)(13). If we think of the problem of dimensionality reduction as one of removing or combining features that are highly correlated, then clustering might be used for reducing dimensionality.

For the purposes of pattern classification the most serious criticism of the approaches to dimensionality reduction that we have mentioned, is that they are overly concerned with faithful representation of the data. Greatest emphasis is usually placed on those features or groups of features that have the greatest variability. But we must keep in mind, for classification, we are interested in discrimination not representation (8).

In the previous sections, we have discussed algorithms to find clusters from a sample of data with a certain distribution. A similar but slightly different problem is to reduce the number of samples while maintaining the structure of the distribution. This may be viewed as selecting a small number of representatives from a distribution. Besides clustering, the reduction of sample size has a number of advantages and applications such as reduced database management and computation complexity.

An important application related to data reduction using clustering comes from the fields of digital signal processing and communication. As a part of coding signals for transmission or storage, the signal values are usually quantized. This one-dimensional quantization is usually done in what is known as an analog to digital (A/D) converter. This is a mapping of a continuous interval, usually speech, music or electromagnetic waveform, to a finite number of levels as in Figure 5.
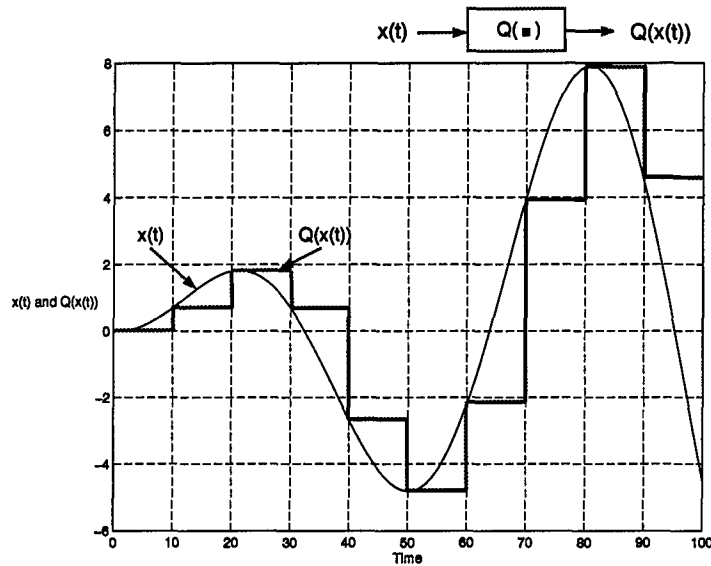
Figure 5. Mapping of continuous interval to finite number of levels

The quantization process may be considered to produce a noisy version of its input due to

quantization error. The error can be quantified in a mean-squared sense:

$$MSE_q = E(x(t) - Q(x(t)))$$ (13)

where MSEq is known as the distortion metric. The scalar quantization, above, is extendible to

higher dimensions and is called vector quantization. Figure 6 depicts this.

Vector quantization has many similarities to pattern recognition clustering but with different

terminology. As in clustering, a similarity metric or distance metric d must be chosen. The

cells are defined implicitly by assigning a vector to the center to which it is nearest. That is,

vector quantization consists of the mapping of vector y into one cell Z_j. When the distance

function is chosen to be the Euclidean distance, then the design algorithm is the k-means algorithm.

Linde, Buzo, and Gray (22) showed that the algorithm work with other possible choice of distance

27
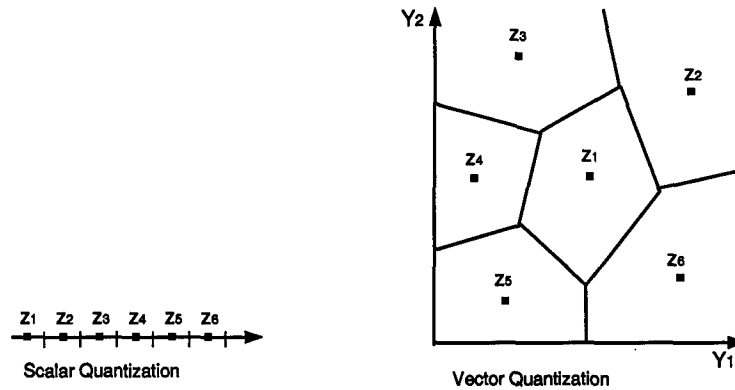
Figure 6. Scalar and Vector Quantization

functions known as the Linde-Buzo-Gray (LBG) algorithm (34). For more detailed review of vector

quantization we refer you to the article by Makhoul, et al. (24).

2.3.3 The Batch k-means Algorithm. Linde, Buzo, and Gray (22) describe the General

Lloyd Algorithm which is used to design a vector quantizer. The General Lloyd Algorithm is

often refered to as the LBG Algorithm and when the distance function is chosen to be Euclidean,

the algorithm is known as the k-means algorithm. The k-means algorithm attempts to minimize

the euclidean distance from all members in a codebook vector to that of the input vector. This

minimization is done through an iterative method which terminates when an average distortion

metric stabilizes (30).

Specifically, the k-means algorithm starts with the choice of the number of cluster centers

k that will represent the distribution. The algorithm then involves the following simple iterative

method: Assume we are given that there are N data samples x^n, and we wish to find a set of k

clusters representative vector u_j where $j = 1, ..., k$. The algorithm tries to cluster the data samples

x^n into k disjoint subsets S_j containing N_j data samples, where each subset represents a cluster.

28

The attempt is to make the samples in the same cluster be somehow more similar than samples in different clusters. One way to make this into a well defined problem is to select a criterion function that measures the clustering quality of any partition of the data. The most widely used criterion function for clustering is the sum-of-squared-error criterion given by (3)

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \| x^n - u_j \|^2 \tag{14}$$

where u_j is the mean of the data sample in set S_j and is given by

$$u_j = \frac{1}{N_j} \sum_{n \in S_j} x^n. \tag{15}$$

The following are the steps in the k-means algorithm (30):

1. Select a stopping criterion which is defined as an acceptable level of total Euclidean distance.

2. Choose the number of codebook vectors k.

3. Using some initialization technique, select the initial locations for the k codebook vectors. It is acceptable to use the location of k samples randomly selected from the design set as the randomly selected initial locations.

4. Calculate the Euclidean distance between each data point and each codebook vector. If the total Euclidean distance is less than the stopping criterion, go to Step 6.

5. Compute codebook membership by determining the closest codebook vector for each data point.

6. Update the location of each codebook vector to the centroid of all the data points that are members of each codebook vector's domain. Go to Step 2.

7. The algorithm has converged and the procedure is terminated.

2.3.4 Kohonen Self Organizing Features Maps. One of the most obvious characteristics of the operation of the human brain, and one of the most central problems in information sciences is the economic representation of data with all their interrelationships. In the subconscious information processing and in thinking, there is a general tendency to compress information by forming reduced representations of the most relevant facts, without the loss of knowledge about their interrelationships. In this self-organizing process, we are aiming at mappings which transform a signal pattern of arbitrary dimensionality onto a one- or two-dimensional array (18).

The Kohonen self-organizing features map, an unsupervised learning algorithm, will serve as an example of the self-organizing process. Since this is an unsupervised algorithm, the data is unlabeled. We will base our treatment of Kohonen self-organizing feature maps on a similar outline as found in Rogers, et al. (29). The Kohonen network is shown in Figure 7.

Before we continue our development, it is assumed that the reader has a basic understanding of artificial neural networks. For the reader that has never encountered neural networks before or for a review of the topic, the reader is refered to Lippmann (23) or Rogers and Kabrisky (29).

Some basic characteristics of the Kohonen net is that it is feed-forward only and consists of an input layer and usually, a two-dimensional output or clustering layer. It mathematically transorms multidimensional input data vectors and maps it to the output vectors. The distribution of the output nodes is spread according to the pdf of the input data. If it is properly trained the nodes of the Kohonen output layer that are spatially close are sensitive to similar inputs. The similarity measure of the input is determined by the similarity metric used during the training phase and weight adjustments. A heuristic for a well trained Kohonen net is that every output node has an
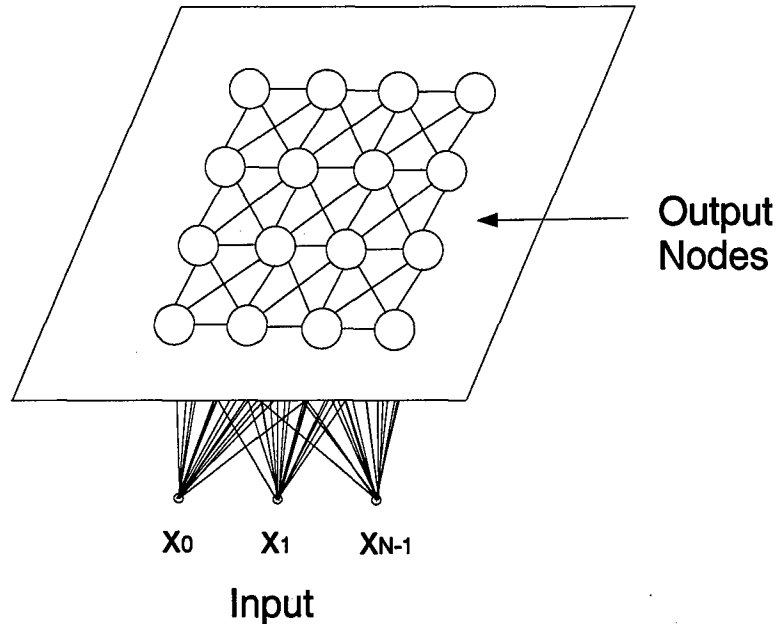
Figure 7. Two-dimensional array of output nodes used to form feature maps. Every input is connected to every output node via a variable connection weight.

equal opportunity of winning or firing. The winning node is the node whose weights are closest or most similar to that of the unlabeled input.

Following are the steps in the Kohonen net (29):

1. Select a learning rate of the Kohonen map for the stopping criterion. For example, the learning rate will be small if the distortion measure of the total map changes little between epochs/trial.

2. Choose the dimension of the output layer (m x n).

3. Initialize the weights to small random value, and the neighborhood size. The weight can be initialized by using a uniform distribution from -0.5 to +0.5 or a potentially better method is to initialize to training exemplars.

31

4. Select both the learning rate by which to adjust the winning codebook vector and also the learning rate for the codebook vector's neighbors. (These can be the same)

5. Randomly select a normalized input vector and compute the distance to each codebook vector. The closest codebook vector to the input vector is the winning node.

6. Conscience (This is an optional step). Check to make sure that the winning node has not already won its share of input vectors for this epoch through the data. If it has, pick the node with the second closest codebook vector to the input vector.

7. Using the learning rate, update the codebook vector of the winning node, moving it closer to the input vector. Similarly, update the nodes of the neighbors using the learning rate selected for neighborhood updates.

8. Repeat step 5 through 7 for all input vectors.

9. Change the learning rates and/or change the neighborhood size and run another epoch. Once the weights of networks are not changing significantly go to Step 10.

10. The algorithm has converged and the procedure is terminated.

One problem commonly noted with the Kohonen algorithm is that it represents a compression of multidimensional data to just one or two dimensions. The goal is to compress information by forming reduced representations of the most relevant facts without loss of knowledge about their interrelationships. The hope is that the projection onto the two dimensional space retains the interrelationships of what is close in the higher dimensional space. Sorry to say but this projection is not unique and the multi-dimensional data representation may be too restrictive of a view. Roger, et al. propose a good analogy of this problem. They refer to it as the flattened- fauna analogy (29). Suppose we had a once three dimensional animal that we have come upon that has

been flattened along the road side. The projection of the three dimensional animal onto the two dimensional roadway is dependent on the position of the animal when it was flattened. You would hope that the foot and ankle would be close together on the flattened version and we could keep their basic physical interrelationships. Unfortunately, too often the ear of the creature, which is not close in physical relationship to the ankle, may end up next to the ankle in the two dimensional representation. This problem gets worse as the dimensionality of the original space increases. This implies that depending in which order the input vectors are presented to the algorithm will determine the end resulting projection. Similarly Rogers, et al. have found that the result of the Kohonen learning algorithm will be different for different initial random weights.

But things are seldom as confused as they are depicted by the flattened-fauna analogy. The actual Kohonen weights are of the same dimensionality of the input data. Therefore, it is only when the weights are projected onto the Kohonen layer that the distortion of the high-dimensional distances occurs.

2.3.5 Clustering Seed. One unique technique for clustering of the HRR radar signature is to use a "Cluster Seed" method. This method was developed by Beckner, et al. (2) in 1995 with good results. The basic concept is the ranking and ordering of signatures in a linear fashion on the basis of signature similarity. One specific signature from the data base is chosen and then compared to the rest of the signatures, and then each of the signatures are placed in a match list in order of their match quality to the specific signature. The signature from the database which is the best match is placed in the first position of the match list. The next-best matching signature is placed in the second position in the match list and so on. The position of a signature in a match list is called its match level, or more simply, its level. The best-matching signature is called a level-zero

match, the next-best match is a level-one match, and so forth. This process of creating a match list is done for all the signatures in the data base.

The cluster formation process is based on data obtained from such a pairwise comparison of signatures in the database. This process is illustrated in Figure 8 and consists of a cluster seed, cluster nucleus construction, and finally the addition of the desired number of outer cluster layers. In Figure 8 the labeled circles represent each individual signature. The lines between signatures represent the matches between signatures. The number associated with these lines is the level of the match. The location of the arrow at the end of the line indicates the signature which is contained in the match list of the signature from which the line originates. An indicated match without a level specifier means that a match of some level exists having a match quality metric value exceeding a specific minimum value. Many other matches may exist beyond what is shown in the figure. The figure is intended to illustrate only the necessary match conditions for constructing cluster seeds, nuclei, and higher level clusters.

The first step in forming a signature cluster is specifying one or more seed signatures. Beckner, et al. chose pairs of signatures which were level-zero matches to each other. These pairs of seed signatures are most similar to each other. An advantage of this seed selection procedure is that it is fully automatic and depends only on the properties of the signatures, and not on arbitrarily-imposed conditions.

The second step in cluster formation is building the nucleus. In the clustering formation process, the nucleus consists of the seed plus all zero-level matches to any member of the nucleus found in the match lists of members of the nucleus. The nucleus is that part of a cluster to which all

A Cluster Seed

A Cluster Nucleus

A Level 1 Cluster
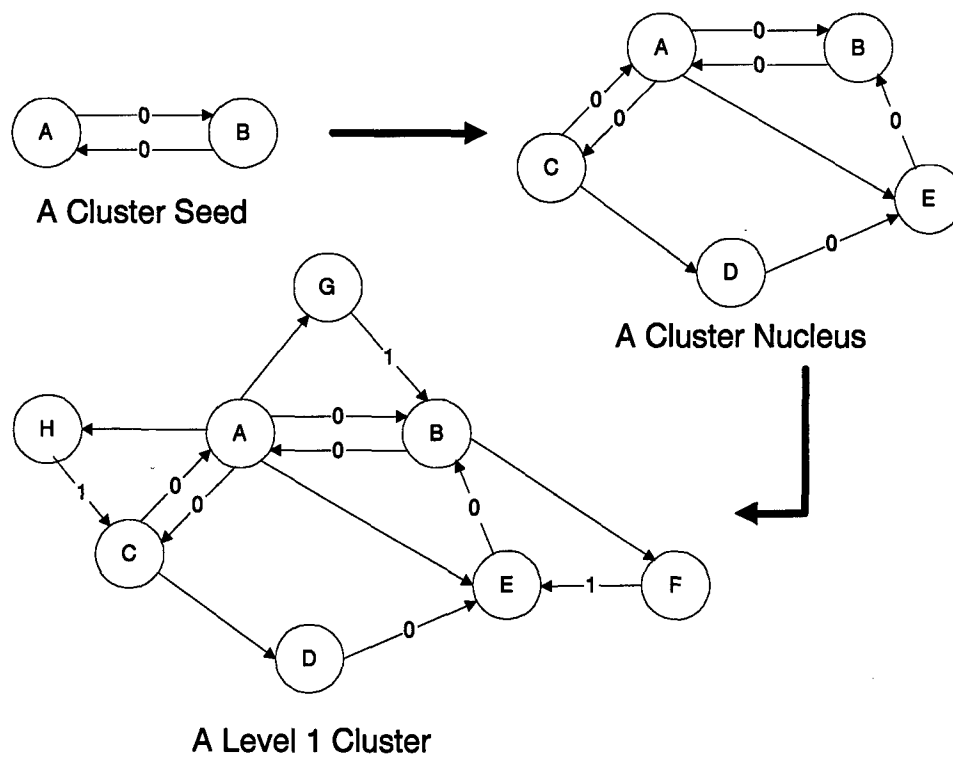
Figure 8. Cluster formation process

cluster members must match and be matched by. Thus one necessary requirement for membership in a cluster is for the signature to be found in the match list of a member of the nucleus.

The final step in cluster formation is adding the signatures which form the outer shells of the cluster. These signatures are those in the match lists of the members of the nucleus and whose match lists contain a match to a member of the nucleus at the match level corresponding to that of the shell, and which have not been assigned to a lower match number shell. These signatures are added to the cluster starting at match level 1 and proceeding to higher match levels. This assures that each cluster member is assigned only to the lowest possible match number shell.

2.4 Chapter Summary

This chapter has presented the theoretical justification for the approaches used in this thesis. The key ideas are summarized as follows:

1. For any statistical pattern recognition problem, the minimum average probability of error is obtained using the Bayes decision rule, which is to assign each sample to the class with the maximum *a posteriori* probability. The error rate associated with this decision rule is called the Bayes error.

2. The Gaussian classifier is a parametric classifier that uses the Bayes decision rule.

3. To achieve Bayes error we assume we know the pdf for all the classes. This assumption is very unlikely and estimation of these density functions is necessary. Two non-parametric pdf estimation methods, Parzen and k-NN, were discussed as well as methods to estimate the Bayes error.

4. Bayes error is difficult to estimate for certain problems and confidence intervals must be used to quantify the results.

5. Two unsupervised clustering algorithms, k-means and Kohonen self organizing feature maps were discussed.

These ideas are brought together in the following chapter, where two distinct approaches for clustering, k-means and Kohonen, are presented.

III. Methodology

3.1 Introduction

Having developed in Chapter II the necessary theoretical concepts necessary to understand the two clustering techniques (k-means and Kohonen), this chapter presents the details of implementation. First, Section 3.2 discusses the general approach followed in the batch k-means clustering algorithm with variations in the ways of initializing the algorithm. Next, Section 3.3 describes the implementation of the general approach of the Kohonen self organizing features maps algorithm. Finally, Section 3.4 details how to implement a Vector Quantizer (VQ) as our classifier using codebooks generated from our two clustering techniques. A flow chart of how to build the codebooks and to implement a VQ procedure is given in Figure 9. In the following sections we will
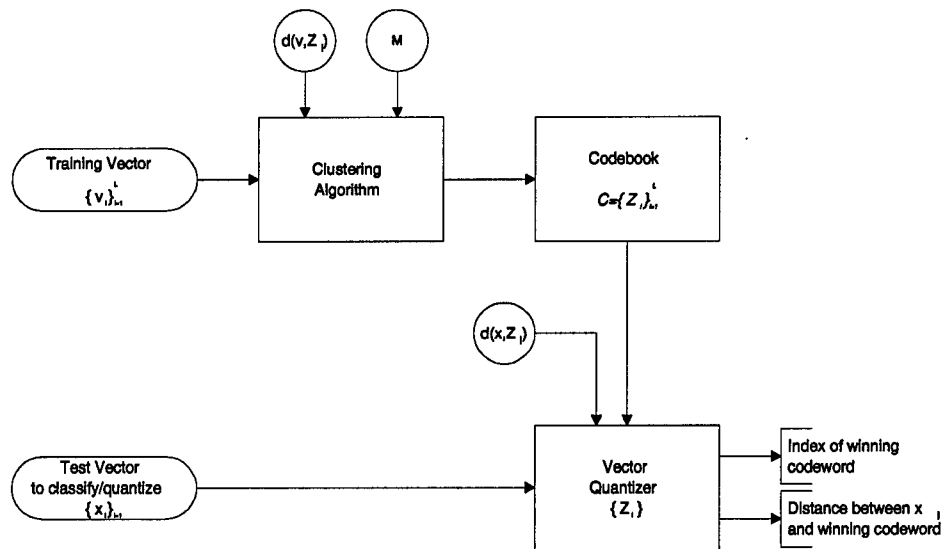
Figure 9. Flow chart of how to build a codebook and to implemant a VQ

go into the details of the different parts that go into our system.

3.2 A General Approach for Batch k-means Clustering

As discussed in Chapter II, the k-means algorithm is based on the generalized Lloyd algorithm. In this section we discuss the implementation of this algorithm that we used. The algorithm described here is implemented in MATLAB and the actual code is contain in Appendix A. Figure 10 is the basic flow diagram for the general approach taken for the k-means algorithm.
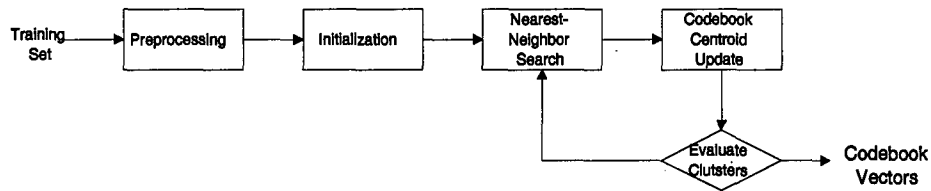
Figure 10. Flow chart of k-mean algorithm

3.2.1 Preprocessing. In the preprocessing step we do any necessary preprocessing of the input data prior to sending it to the clustering algorithm. This step would include registration, normalization, feature saliency, down sampling, noise reduction, weighting, and etc. For some problems, registration or alignment is done to insure the properly corresponding features are compared to each other and that misalignment does not influence our results. Normalization of the input vector is performed to ensure all inputs are within the same relative magnitude range. Either statistical normalization or energy normalization can be used, depending on the problem. Feature saliency and down sampling should also be accomplished at this stage. To avoid the curse of dimensionality, we should examine only the relevant features and deal with the smallest dimensionality problem possible. Noise reduction is an important step to insure that input has only minimal corruption, either through filtering or sampling and subtraction of the noise. We can also weight our features

39

to reflect their importance or their risks. This list of preprocessing steps is by no means exhaustive and other important steps should be taken depending on the problem at hand.

3.2.2 Initialization. A major limitation of k-means clustering is that the location of final codebook vectors may be influenced by how we set the initial positions of the codebook vectors and the order of presentation of the training samples, which will ultimately belong to different codebook vectors. The proper choice of codebook initialization scheme is key in avoiding this problem. A variety of codebook initialization schemes have been developed in hope of providing accelerated convergence, achieving better clusters that more accurately represent the distribution of the data points and provide the flexibility in the number of cluster centers needed to represent the data. Linde, et al. (22) suggests a splitting method (LBG algorithm), whereby the k-means algorithm is applied at each power of two (giving codebook sizes of 1, 2, 4, 8, ...). Katsavounidis, et al. (17) proposed a maxi-min method, while recently, DeSimio, et al. (5) proposed a Karhunen-Loeve initialization (KLI) scheme, whereby the cluster centers are placed along the principal component axes of the training data's covariance matrix. For the purpose of this thesis we will limit our study to the latter two techniques, the maxi-min and the KLI.

3.2.2.1 First-k-samples. The most basic of all initialization schemes is to use the first k samples of the training data. With this scheme one uses the location of the first k samples as the selected initial locations for the codebook vectors. This scheme is very simple to implement, but is very inefficient and can cause poor representation of the data's distribution, since the first k samples might all belong to the same cluster (36).

A possible improvement to this basic initialization scheme is to break the data set into subsets and then average the samples of each subset to get a mean representation of each subset. For

40

example, given that we have a 1000 samples and we wanted 10 codewords or clusters. What we could do is take the first 100 samples and find a mean representation for those samples and use that as our first codeword initial location. We would then take the second subset of 100 samples and find the second codeword initial location and so on. This would hopefully give a better representation of the samples set's distribution.

3.2.2.2 Maxi-min Initialization. Katsavounidis, et al. (17) proposed the maxi-min initialization algorithm based on the idea that widely separated data points are likely to belong to different classes. Let x_i, $i = 1, ..., N$ be the sequence of training vectors. The maxi-min initialization procedure is as follows:

1. For the training set, determine the norms for all the training vectors. Pick the vector with the largest norm as the first codeword in our codebook.

2. Using our symmetry metric (Euclidean distance in our case), compute the distance from the first codeword to all of the remaining training vectors. Choose the training vector with the largest distance from the first codeword as the second codeword. Now, we have a codebook with two codewords.

3. Generally, when we have codebook size n, $n = 2, 3, ...$, we would compute the distances from all the codewords in our codebook to all remaining training vectors x_i in our training set. The smallest distance between each training vector, x_i, and all the codewords will be used as the distance between x_i and the codebook. Then the training vector with the largest distance will be chosen to be the $(n + 1)$th codeword. We repeat this process until we obtain the final codebook size, M.

3.2.2.3 Karhunen-Loeve Initialization. DeSimio, et al. (5) proposed the Karhunen-Loeve based initialization (KLI) algorithm, where cluster centers are placed along the principal component axes of the training data's covariance matrix. The idea behind this method is borrowed from the Karhunen-Loeve transform to position initial codewords along the directions of maximum variance. Thus, more codewords are used in the directions where the data have relatively large variances versus directions where the variances are small.

The Karhunen-Loeve initialization procedure is as follows:

1. The first step is to specify the number of codewords to be developed.

2. The covariance matrix, W, of the k-dimensional training data is then computed along with the associated eigenvalues, λ_i, and eigenvectors, v_i.

3. The number of codewords along each eigenvector direction is found according to

$$N_i = \left\lfloor \frac{\lambda_i}{\sum_{j=1}^{k} \lambda_j} + 0.5 \right\rfloor N_c, \tag{16}$$

where N_c is the total number of codewords. Codewords are formed by scaling the eigenvectors such that the N_i codewords are uniformly distributed over $\pm 2\sigma_i$.

DeSimio, et al. have shown that for certain data sets, the KLI codebook initialization method results in codebooks with lower mean-square-error than codebooks generated with either the LBG or the Maxi-min algorithms.

3.2.3 Nearest-Neighbor Search. The search for the nearest-neighbor is where, for each training vector, we find the closest codeword and assign the training vector to that codeword.

3.2.4 Codebook Centroid Update.   There are two ways for updating codeword centers; the batch method, where the update takes place after all the data points are assigned to the distinct codeword centers, and the sequential approach, where the codewords are updated after each sample is added to a codeword. In the sequential approach, the initial centers are chosen by some initialization method mentioned in Section 3.2. Then, as each data sample x_i is presented, the nearest codeword mean, μ_j, is updated using the learning rate η and the following equation (3)

$$\Delta\mu_j = \eta(x_i - \mu_j) \tag{17}$$

In the batch version of the k-mean algorithm the initial centers are chosen by some initialization method. Then each data sample is re-assigned to a new codeword based on which ever codeword is the nearest in terms of the Euclidean distance. The mean μ_j of the codewords and their new members are then recomputed.

3.2.5 Evaluate Clusters.   The procedures in Sections 3.2.3 and 3.2.4 are repeated until there are no further changes in the grouping of the data samples.

3.3 A General Approach for Kohonen Self Organizing Feature Maps

As discussed in Chapter II, the Kohonen self organizing features maps algorithm is a neural network based on the idea that self organizing maps learn to recognize groups of similar input vectors in such a way that output nodes physically close together in the Kohonen layer respond to similar input vectors. In this section, we discuss the general approach taken in this thesis to implement this algorithm. The algorithm is implemented in MATLAB using the Neural Network Toolbox. The basic theory comes from Kohonen's book titled, "Self-Organization and Associative

Memory" (18). Figure 11 is the basic flow diagram for the general approach taken for the Kohonen algorithm.

```
Training ──→ ┌──────────────┐ ──→ ┌──────────────┐ ──→ ┌──────────────┐ ──→ Codebook
Set          │ Preprocessing│     │ Initialization│    │   Training/  │     Vectors
             └──────────────┘     └──────────────┘     │   Mapping    │
                                                        └──────────────┘
```
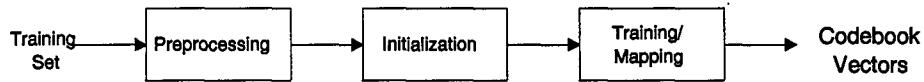
Figure 11. Flow chart of Kohonen process

3.3.1 Preprocessing. As with the k-means algorithm, the preprocessing step is where we would do any necessary preprocessing of the input data prior to sending it to the clustering algorithm. This step could include registration, normalization, feature saliency, down sampling, noise reduction, weighting, and etc.

3.3.2 Initialization. Lets discuss the intialization steps and some of the intricacies of this algorithm. First, we must come up with a learning rate, η, which is used to update our weights. Commonly, use a starting η value of about ≤ 1, which is then linearly reduced. We will use the same learning rate to update both the winning node and its neighbors. Second, we choose the size of our output layer. The output nodes dimensions can either be one-dimensional or two-dimensional. Third, we choose a symmetry metric; we use Euclidean distance. Last, we decide the number of epochs to run so that the weights of the networks have a chance to stabilize and converge.

As with other learning algorithms we must initialize the weights of our neural network to small random numbers. This algorithm will eventually assign weights that approximate our input data. Therefore, we should initialize our weights to values in the range of our input data or normalize the data to the range of our weights. For our algorithm, we will take the minimum and maximum expected values of our inputs and then take the median values of those inputs as our initial weights.

3.3.3 Training. Self-organizing maps differ from conventional neural network learning in terms of which nodes get their weights updated. Instead of updating only the winner, feature maps update the weights of the winning node and its neighbors. The result is that neighboring output nodes tend to have similar weight vectors and are responsive to similar input vectors.

Beginning with our initial learning rate and a neighborhood size large enough to include all the nodes, we use the following weight update equation:

$$w_{ij}^+ = w_{ij}^- + \eta(x_i - w_{ij}^-), \tag{18}$$

where i is the index for the input, j is the index for the Kohonen nodes, w_{ij}^+ is updated weight, and w_{ij}^- is the old weight. During training, both the neighborhood size and the learning rate are decreased. Thus, the node's weight vectors initially take large steps all together toward the area of input space where the input vectors are occurring. Then, as the neighborhood size decreases to 1, the map tends to order itself topologically over the presented input vectors. Once the neighborhood size is 1, the network should be fairly well ordered and the learning rate is slowly decreased over a longer period to give the nodes time to spread out evenly across the input vectors.

The Kohonen self organizing features map will order themselves with approximately equal distances between them if input vectors appeared with even probability throughout a section of the input space. But if input vectors occured with varying frequencies throughout the input space, the features map layer will tend to allocate nodes to an area in proportion to the frequency of input vectors there. Thus, the features maps, while learning to categorize their inputs, also learn both the topology and distribution of their inputs.

3.4   The Classifier

The classifier we have chosen to use is a Vector Quantizer (VQ). In this section we will describe the actual implementation of a VQ for classification. The following are the steps to implement our VQ classifier:

1. Assume we have M classes. Create a codebook for each class $CB_1, ..., CB_M$.

2. For each input vector, x_i, compare each of its features with each codeword's features and find the distance metric d_{min} for all codewords in all codebooks. For instance if we had a c dimensional input vector, we would have c distance metrics for each codeword comparison.

3. Accumulate or sum all the distance metrics for each codeword to compute the total distortion, D_i.

4. Choose the codeword with the minimum value of total distortion, D_i. The class j^* that owns the codebook with the winning codeword is declared as the class for that input vector. The VQ then outputs the winning codeword and associated minimum total distortion.

These steps are illustrated in Figure 12. This approach has been applied to the Speaker Identification problem with very good results. $P(error) \approx 2\%$ for 20 speakers using Mel-Frequency Cepstral Coefficient (MFCC) as features.

3.5   Chapter Summary

The methodology used in each of the k-means and Kohonen clustering algorithms was presented. With each clustering algorithm, codebooks are generated and then used in a vector quantizer classifier for classification. As we will see, each method involves finding some of the optimum parameters for the particular problem.

46
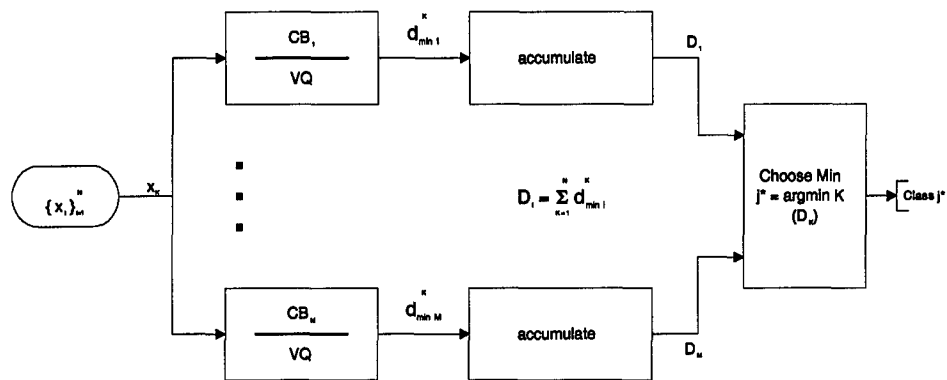
Figure 12. Flow Chart of the VQ

The following chapters present results obtained by applying these techniques to several pattern recognition problems. Chapter IV presents the results obtained by the two clustering techniques for several toy problems. These toy problems allow for a better understanding of the results from these techniques. Chapter V presents the clustering and classification performances for the HRR problem obtained by each of the two methods.

IV. Analysis and Results of Toy Problem

4.1 Introduction

Before we apply the clustering techniques and algorithms described in the previous chapters to the HRR target identification problem, each of these techniques are applied to several test problems. These test problems use low dimensional training samples with known distributions so we can easily draw proper inferences on the effectiveness of the techniques and our algorithms.

The first set of test problems illustrates how well the k-means clustering technique works. We first look at how initialization, in particular the KLI and maxi-min methods, effects the final results of a k-means algorithm. Next, we used XOR data with the k-means algorithm to see if it will create proper codewords to represent the training data's distribution. The XOR data can be created by random generation of four Gaussian distributions with means of (1,1), (1,-1), (-1,1), and (-1,-1) and with equal or varying variances.

The second set of test problems illustrates how well the Kohonen self-organizing features map technique works. We investigate some of the basic intricacies of the algorithm. We determine how well the distribution of the nodes is spread according to the pdf of the data. In a correctly trained Kohonen layer, nodes that are spatially close, close in location on the Kohonen layer, are sensitive to similar inputs. We then examine a one dimensional and a two dimensional test case.

4.2 k-means

4.2.1 Description of Test Case on Initializations. To provide an initial set of problems that are easily visualized and analyzed, we use four groupings (250 data samples in each grouping) of two-dimensional data that have Gaussian distributions. These four groups of training data are

48

illustrated in Figure 13. Using this data, we apply the KLI and maxi-min initialization techniques to get 20 initial clustering centers for use with our k-means algorithms, the results are shown in Figure 13. The "+" represents the location of the clustering centers. The KLI technique disperses the desired number of cluster centers along the principal component axes of the data's covariance matrix. Note the cluster centers for the KLI are not placed near the data as in the maxi-min methods. As for the maxi-min technique, it disperses the desired number of cluster centers by taking data samples that are maximumly spaced from each other as the initial cluster centers.

After initializing our cluster centers, the k-mean algorithm determines the final cluster center locations. The final cluster center locations are shown in Figure 14. Visually, both initialization techniques seem to create proper codewords to represent the training data's distribution. It was reported by Desimio et al. that the KLI technique provided lower MSE values and require less computational effort (5). But, it is well known that the MSE does not imply highest classification accuracy or highest perceived visual quality (8).

4.2.2 Description of Test Case using XOR Data. We will run the same test as in section 4.2.1 using the XOR data set as our training samples. The desired number of codewords is eight for this test case. The results are shown in Figure 15 and Figure 16. The initial cluster centers are in the proper location as to agree with the theory for each of the initialization techniques. Interestingly, there are only five final cluster centers for the KLI initialization technique. This is because three of the initial clustering centers gained no members when they were subject to the k-means algorithm and are not used. Since the KLI technique places the initial cluster centers along the principal component axes of the data's covariance matrix and not near the data samples,

49

Figure 13. Initialization of k-means algorithm using KLI vs. maxi-min for two-dimensional data
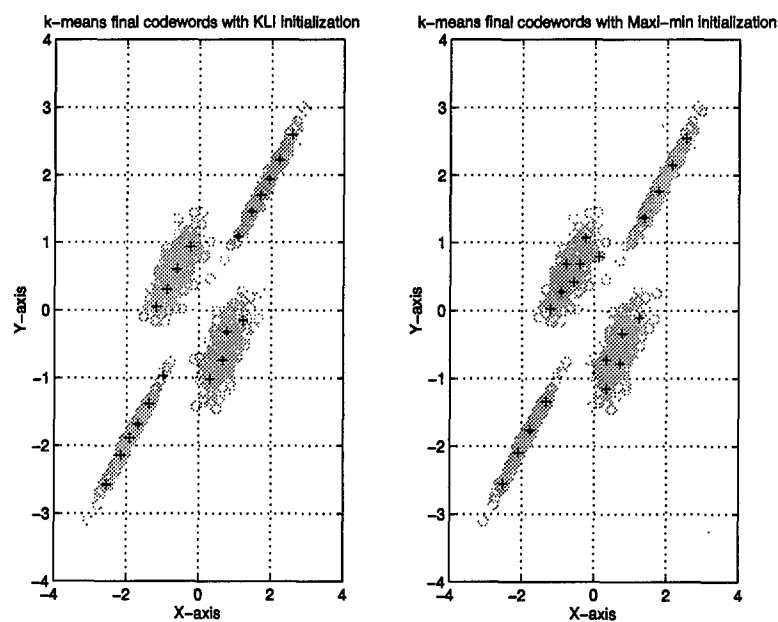
Figure 14. Initialization of k-means algorithm using KLI vs. maxi-min final cluster center locations for two-dimensional data

some codewords may not gain any members. In this case the maxi-min has a lower MSE than that of the KLI, and the KLI technique failed to achieve our desire number of clusters.

4.3 Kohonen

4.3.1 One-dimensional Test Case.   To demonstrate the abilities of the Kohonen self-organizing map we first look at how well it can map onto a one-dimensional output layer. We first create a data set of a simple sinusoidal function which is represented by 150 two-element unit input vectors. The sample set is shown in Figure 17. We then ran our Kohonen algorithm for 1500 cycles (epochs) using a one-dimensional layer of 15 (1x15) neurons to create a self-organizing map of our input vectors. The resulting map is shown in Figure 18. After 1500 training epochs, the layer had adjusted its weights so that each output node responded strongly to a region of the input space occupied by input vectors. The placement of neighboring nodes' weight vectors also reflects the topology and distribution of the input vectors. Note though, that self-organizing maps are trained with input vectors in a random order, so starting with the same initial weight values does not guarantee identical training results, see Figure 19.

4.3.2 Two-dimensional Test Case.   Now that we have seen how the Kohonen can map onto a one-dimensional output layer, lets look at mapping onto a two-dimensional output layer. For this problem we start by generating 1000 input vectors uniformly distributed as shown in Figure 20. A two-dimensional, five nodes by five nodes map, of 25 nodes is used to classify these input vectors. We then ran our Kohonen algorithm for 5000 cycles (epochs) using the two-dimentional layer of 25 neurons to create a self-organizing map of our input vectors. The resulting map is shown in Figure 21. After 5000 training epochs, the output layer has once again adjusted its weights so
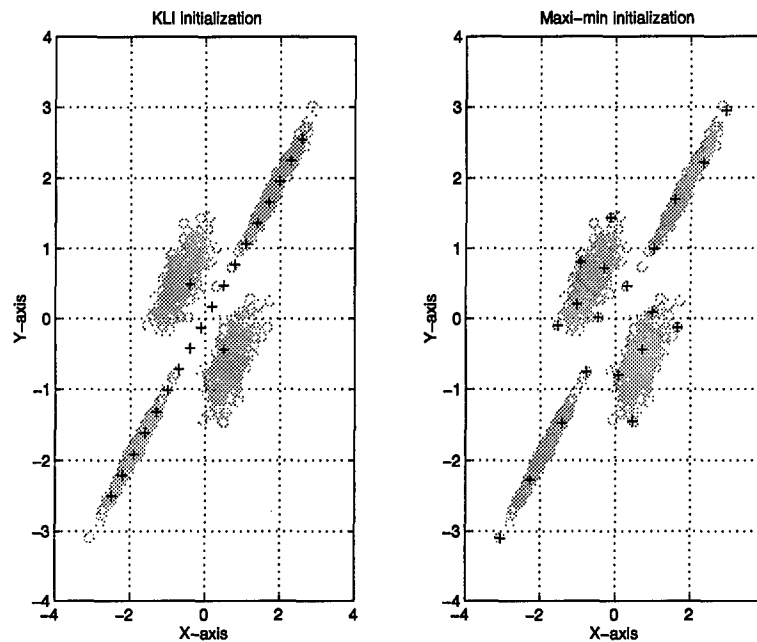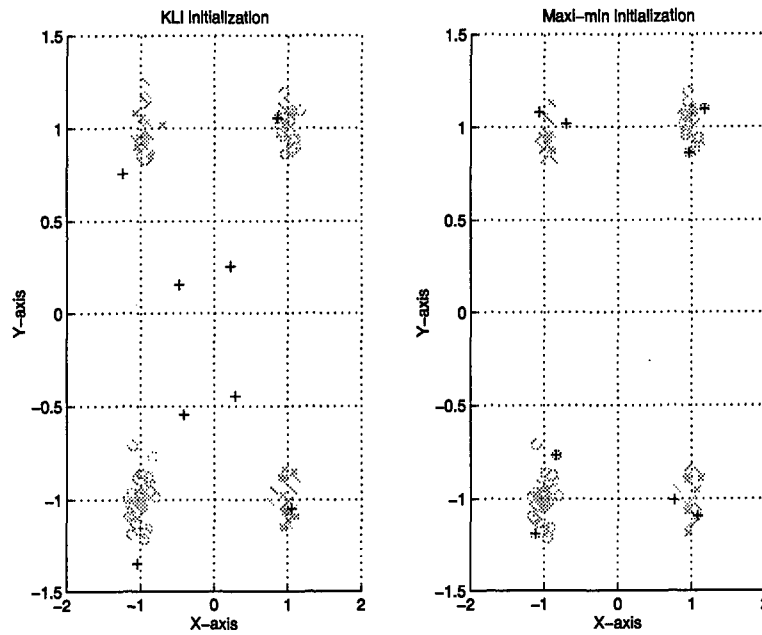
Figure 15. Initialization of k-means algorithm using KLI vs. maxi-min for XOR data
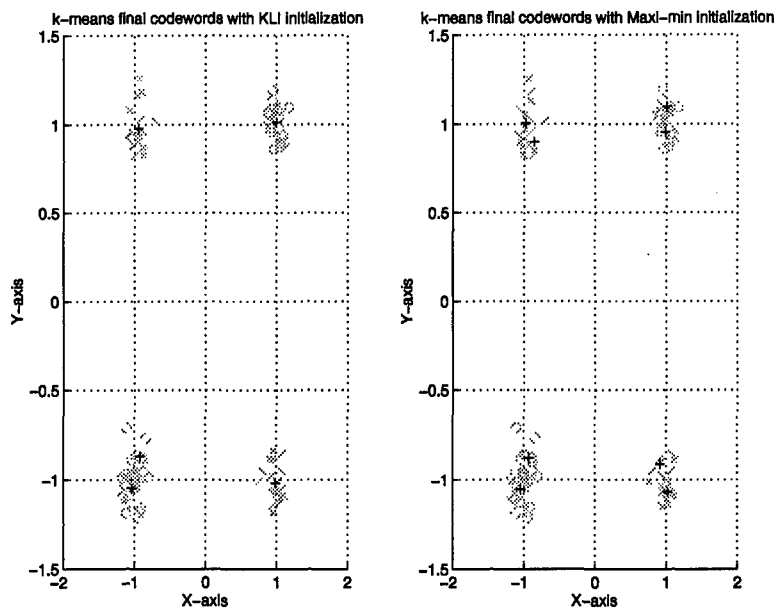
Figure 16. Initialization of k-means algorithm using KLI vs. maxi-min final cluster center locations for XOR data
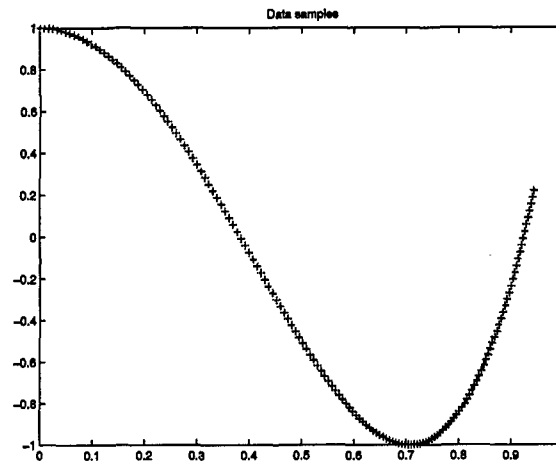
Figure 17. Plot of 1-D data set
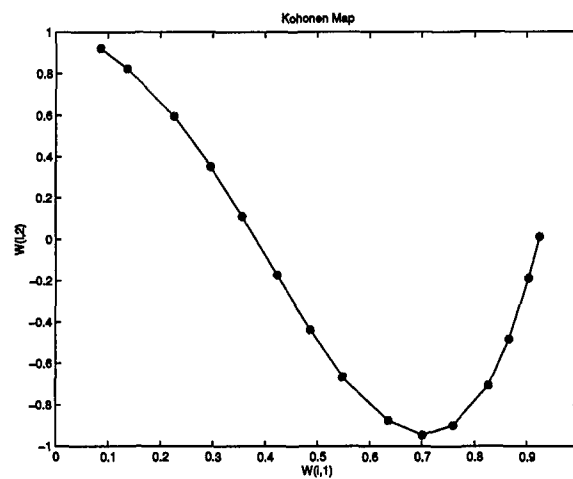
Figure 18. Kohonen self-organizing feature map of 1-D data

53

that each output node responds strongly to a region of the input space occupied by input vectors. The placement of neighboring nodes' weight vectors also reflects the topology and distribution of the input vectors.

The choice to use a 5X5 output nodes as the two-dimensional map of 25 nodes was arbitrary. We could have just as easily used a 1X25 output nodes to create our mapping. The same experiment using a 1X25 nodes output layer is done and shown in Figure 22. Both output layer schemes seem to reflect the topology and distribution of the input vectors well, namely evenly distributed throughout the region. Even though they have different final locations for the output nodes, both are good mappings of the input vectors. Still, it is well known the highest perceived mapping in one- or two-dimensional space of a higher dimensional input vector does not imply highest classification accuracy or best representation of the data as was earlier illustrated with the "Flatten Fauna" analogy, see Figures 23 and 24 (29).

4.4   Chapter Summary

This chapter has shown the results obtained by applying each of the two clustering techniques to a few simple test cases. In general, these results indicate the two techniques worked in general as expected. The end results of each are very dependent on the initial conditions and parameters. These clustering techniques must be augmented with some *a priori* knowledge of the problem to come up with good initial parameters to achieve optimal end results. The following chapter presents the results obtained by applying each of two techniques to the HRR target identification problem stated earlier.
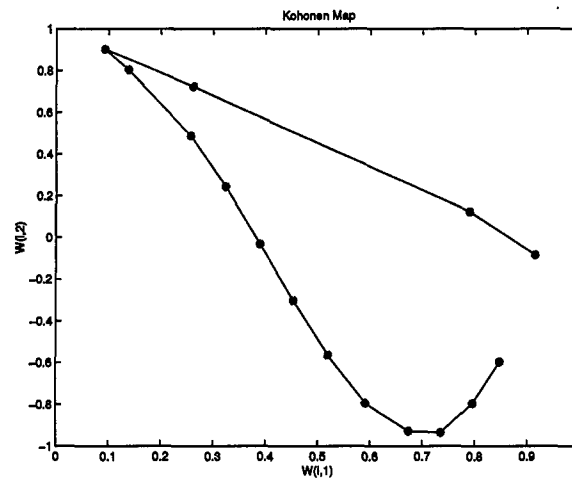
Figure 19. Kohonen self-organizing feature map of 1-D data
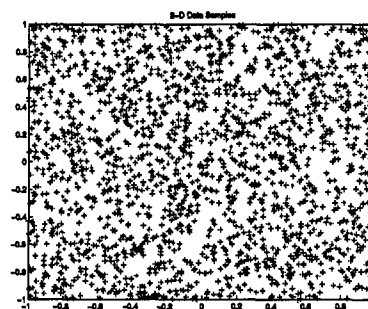
Figure 20. Plot of 2-D data set
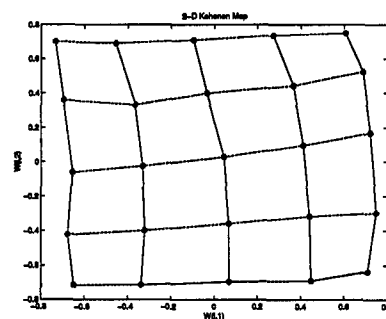
Figure 21. Kohonen self-organizing feature map of 2-D data using 5X5 output nodes
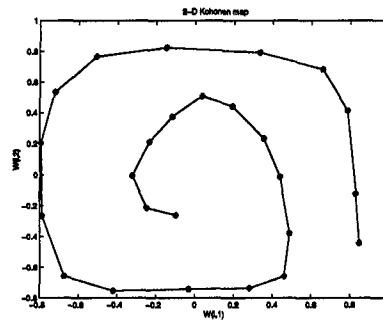
Figure 22. Kohonen self-organizing feature map of 2-D data using 1X25 output nodes
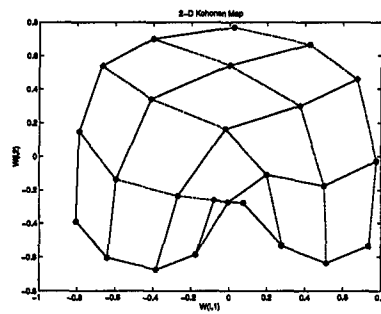
Figure 23. Kohonen self-organizing feature map of 2-D data using 5X5 output nodes
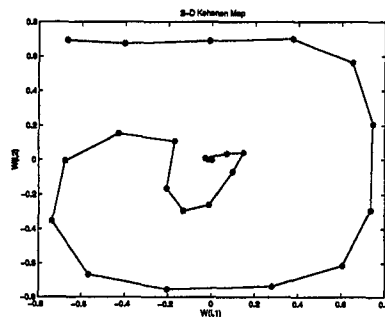
Figure 24. Kohonen self-organizing feature map of 2-D data using 1X25 output nodes

V. Clustering of HRR Data

5.1 Introduction

After validating the two algorithms with the toy problems, the two clustering methods are applied to the HRR target identification problem. This chapter presents the classification results of each of the two methods on a three class problem. Section 5.2 describes the choice of classifier, the data sets used and the initial processing done to prepare the samples for the two clustering techniques. Section 5.3 presents the baseline template creation methods used by our classifier. Section 5.4 and Section 5.5 shows the results of creating templates using the k-means and Kohonen methods, respectively. In Section 5.6, we present the classification results of each of the techniques, compare the results with each other, and discuss some of the implications of the results. Finally, Section 5.7 describes and discusses the data reduction experiment and some of its implications.

5.2 Problem Description

As described in Chapter I, there are two main thrusts in our investigation. First, we investigate the use of clustering techniques to increase the accuracy in constructing signature templates for improved performance in template matching algorithms. Second, we investigate the potential of reducing the total number of required templates per class that must be stored in the database.

5.2.1 Data Preparation. The majority of pattern recognition applications require the original input vectors $x_1, ..., x_n$ to be transformed by some form of pre-processing to give a new set of variables $\tilde{x}_1, ..., \tilde{x}_n$. There are often great benefits in preprocessing and, in the following sections, we discuss the necessary preprocessing in our process.

57

5.2.1.1 The Data.   As described in Chapter I, the HRR range profiles represent the return of a narrow pulse radar signal, as resolved along the radar-target vector. Energy from scattering centers at a given range is put into a range bin corresponding to an incremental distance from the radar receiver. In this analysis, an unprocessed HRR radar signature consists of 812 range bins. Each range bin has in-phase and quadrature components. These HRR range profiles can either be measured or synthetically produced using computational electromagnetic techniques and codes.

The training data for this problem consists of synthetically produced range profiles of three aircraft taken at 1 degree increments in a 31 X 36 degrees aspect angle window, for a total of 1116 range profile per aircraft. The aspect window incorporated 0 degrees to 31 degrees in azimuth and 5 degrees to -30 degrees in elevation. The aircraft are represented by a CAD file with 10,000 to 80,000 triangular facets. A computer code, XPATCH, based on the shooting and bouncing ray technique is employed to calculate the polarimetric radar return and its range profile from the faceted aircraft. Both the first-bounce Physical Optics plus the Physical Theory of Diffraction contributions and the multi-bounce ray contribution are included (1). The three aircraft are identified as targets A, B, and C.

The test data for this problem consists of collected measured data from a ground instrumentation radar on the same three aircraft used in training. For each of the three aircraft, range profiles are collected at eighteen 5 X 5 degrees windows covering 30 X 15 degrees total aspect. The aspect window incorporated 1 degrees to 30 degrees in azimuth and 5 degrees to -9 degrees in elevation. We have approximately 5000 signatures per target, for target A we have a total of 6904 signatures; for target B we have a total of 4715 signatures, and for target C we have a total of 4809 signatures.

Since this is collected on an aircraft in flight, the number of signatures at different aspect angles will not be uniformly distributed, it will vary dramatically from a high of 922 signatures to a low of 1 signature per 5x5 aspect window.

Research in the area of HRR for target identification in the past has been trained and tested on synthetic profiles or trained and tested on measured profiles (26)(10)(7)(20). These two cases are interesting from an academic point of view but have relatively little impact on the real life problem of target identification. In the real world we must have enough data to properly train our classifier, so we must use synthetically generated data, but, we must test on data that we would actually have to classify, which is measured.

Recent increases in the accuracy of electromagnetic predictions and shorter signature prediction time, have made training on synthetic data a feasible alternative in classifier design. The major advantages to using synthetic data over measured data are that the synthetic data is already aligned, a sufficient amount of data for training can be produced, and the task of synthetically predicting all aspect training data is lower in costs and more easily done. The major disadvantage of using synthetic data is that inaccuracies in either the electromagnetic prediction process or the geometry model can result in poor and inaccurate range profiles.

5.2.1.2 Magnitude and Dimensionality Reduction.   As mention in Section 2.3.2 we want to minimize the effect of the "curse of dimensionality". Let x represents the 812 dimensional input vector, with complex value of the return in the nth range bin represented by $x(n)$. Each range bin contains in-phase and quadrature-phase components $x_I(n)$ and $x_Q(n)$. The first step in our preprocessing is to take a simple magnitude for all n.

$$x_{mag}(n) = \sqrt{x_I{}^2(n) + x_Q{}^2(n)} \qquad (19)$$

The resulting input vector is then decimated by four. What we mean by decimate in this context is resampling the data a lower rate after low-pass filtering. The resulting vector y is four times shorter and has a new length of 203 range bins.

5.2.1.3 Normalization. Because the relevant classification information in the HRR range profile is believed to be in the relative magnitudes and locations of the peaks, any normalization or transformation must not impact or skew this information (6). Martin reported that if we normalize on a feature by feature basis we would effect the relative magnitude of the peaks and would inadvertently lose information and overall classification performance (26). Feature by feature normalization is done by first finding the mean and variance of each feature over all the training samples. Then we normalize each sample by subtracting the mean from each feature and dividing by the standard deviation. By doing this we transform each feature to a random variable with zero mean and unit variance.

The proper normalization in order to preserve the intrinsic information in the feature vector is to energy normalize each sample. This is done by first summing the squared value of each range bin and then dividing that value by the total number of range bins. This value is then squared rooted to get the root-mean-squared (rms) energy. The final step is to divide the value of each range bin by the rms energy in the profile so that the rms energy of each profile is unity. This can be expressed mathematically as follows:

$$E = \sqrt{\frac{\sum (y(n))^2}{203}} \qquad (20)$$

$$z(n) = \frac{y(n)}{E} \qquad (21)$$

By energy normalizing, the relative magnitudes of the various peaks are maintained within each profile. By using the energy based normalizing procedure we hope to maintain the necessary intrinsic information needed for classification.

5.2.1.4 Alignment. Feature alignment is a necessity for most pattern recognition systems. A definition of a good feature would be one that is present in the same location across most of the measurements. This is especially true for the case of HRR radar signatures for target classification. In the HRR applications, however, the starting location for all of the features varies from measurement to measurement. It is almost guaranteed that the electromagnetic scatter from aircraft radar return will not always be properly windowed in the range bins. Therefore, achieving proper alignment will be crucial for good algorithm performance. The effects of miss-aligned signature can have drastic impact on algorithm performance, especially when using a template based algorithm. Mitchell performed an alignment study for HRR and showed that 82 percent of the misclassification can be attributed to miss-aligned signatures (27).

To align our signatures we will use a signature to signature or signature to template correlation approach. To find the cross correlation $\phi_{xy}[n]$ between the input vector $x[n]$ and the template $y[n]$ we will use the following equation:

$$\phi_{xy}[n] = \sum_{m=-\infty}^{+\infty} x[m+n]y[m] \qquad (22)$$

61

where n is the shift value. Finding the largest cross correlation value and circularly shifting the input vector by that value we should attain proper alignment between the input vector and the template.

5.2.2 Classifier. For our investigation we will look at two possible classifiers. The first of the two classifier we will look at is a Gaussian classifier known as the Adaptive Gaussian Classifier (AGC). The AGC is currently being pursued in the HRR radar problem at Wright Laboratory. The second classifier that will be used is the Vector Quantizer (VQ). The VQ has shown great performance in the area of speech recognition and show great promise for the HRR problem.

5.2.2.1 Adaptive Gaussian Classifier. The Adaptive Gaussian Classifier is a parametric Gaussian classifier as described in 2.2.3 As mentioned in Section 2.2.4, parametric methods involve the assumption of a specific functional form for the pdf. In this instant we must assume that the data is Gaussian in form. The AGC takes HRR radar signatures, preprocesses them and applies a Gaussian discriminant. We want to look at this classifier since it is similar to the one currently in use at Wright Laboratories and designed by Hughes Aircraft Company.

The AGC is described and called "adaptive" because of the preprocessing steps that are needed to be applied to the unknown input vectors. First, the magnitude of each input vector is taken from the in-phase and quadrature phase components. Then each input is down-sampled and a coarse alignment is done using a circular centroid (CC) approach. The next step is to power normalize the signal and finally do a power transformation prior to sending it to the Gaussian classifier. The power transformation is needed since we are assuming that the data is Gaussian in form even though most range profiles measured form radar systems are not Gaussian distributed. Therefore, we must make the underlying HRR radar signature pdf more closely appear Gaussian.

The power transformation simply involves raising each range bin value to some predetermined constant. The constant is usually determined experimentally based upon the data to be classified.

The Gaussian discriminant function used in the AGC is also different from that which we described in Section 2.2.3. The AGC does not use the full covariance of the features (range bins) data, but estimates only the variance. This is necessary due to the enormous amount of training data necessary to accurately estimate a covariance matrix with d(d+1)/2 parameters, where d = 203. Although, this will give us suboptimal performance, studies have shown that classification accuracy using only the variances performed marginally lower than those using the full covariance (27). Therefore, assuming equal *a priori* probabilities for all classes, and diagonal covariances for each class the discriminant function reduces to

$$g(x) = -\frac{(x - \mu_n)^2}{\sigma_n^2} - 2\ln(\sum_{j=1}^{203} \sigma_n^j) \qquad (23)$$

where the classifier makes a class assignment for each unknown feature vector by picking the largest output from the discriminant function. For a more thorough description of the AGC we refer you to Eisenbies (10).

5.2.2.2 Vector Quantizer. The VQ is a non-parametric classifier. As mentioned in Section 2.2.4, non-paremetric methods makes no assumptions of the data's pdf, but allows the form of the density to be determined entirely by the data. The non-parametric VQ classifier has some very nice intrinsic advantages over that of the AGC.

5.3 Windows using 5x5 Averaging

Identification of aircraft from HRR radar range profiles requires a database of information capturing the variabilities of the individual range profiles as a function of viewing aspect. This database can be a collection of individual signatures or a collection of averaged signatures distributed over the region of viewing aspects of interest. An efficient database is one which captures the intrinsic variabilities of the HRR signatures without either excessive redundancies typical of single-signature databases, or without loss of information common when averaging arbitrary groups of signatures into templates.

One way to prevent over-characterization in our data base is to use a template approach over that of single-signature databases. The easiest way to form templates is with the azimuth/elevation grouping of range profiles. These signatures are grouped into small sets which span a specified azimuth and elevation area, see Figure 25. The azimuth/elevation windows are grouped in such a manner as to hopefully guarantee stationarity in the measured range profiles making up the window. For our investigation we will use 5 degrees by 5 degrees template regions to create each template for our baseline results with the VQ and the AGC. Over our total aspect window of interests this would result in 42 total templates per class.

5.4 Batch k-means

To avoid under-characterization in our database, where there is loss of information that is common when averaging arbitrary groups of signatures, we will cluster the signatures to get better representation. We will first try the k-means algorithm with three different initialization schemes.

5.4.1 5x5 Averaging Initialization. Since we are using 5x5 degrees window templates for our baseline results, these templates might serve as good initial codewords location for our k-means algorithm. The results are shown in Figure 26. The figure shows the total aspect window of interest for all three classes. The different shading shows a particular cluster that each individual aspect would belong to. We have a total of 42 clusters since this is the amount of clusters used with a 5x5 azimuth/elevation window method.

5.4.2 Maxi-min Initialization. The results of clustering with a k-means algorithm using the maxi-min initialization scheme is shown in Figure 27.

5.4.3 KLI Initialization. The results of clustering with a k-means algorithm using the KLI initialization scheme is shown in Figure 28.

5.5 Kohonen SOFM

As with k-means, to avoid under-characterization in our database, we will now try the Kohonen algorithm with a two different output nodes structures.

5.5.1 Output nodes structures. The results of clustering with a Kohonen algorithm using a 6x7 output nodes structure is shown in Figure 29. The results of clustering with a Kohonen algorithm using a 1X42 output nodes structure is shown in Figure 30.

5.6 Clustering Results analysis

Several observations are immediately apparent upon inspection of the plots in Figures 17, 18, 19, 20, and 21. First, the cluster are not nicely grouped in 5x5 azimuth and elevation win-

SECTORING
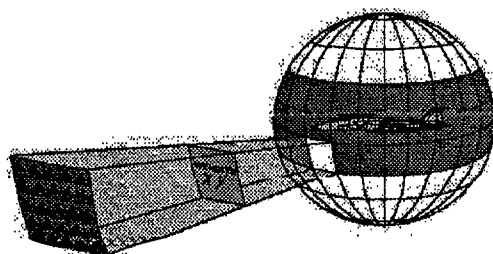
Figure 25. Aircraft mxn window sectoring
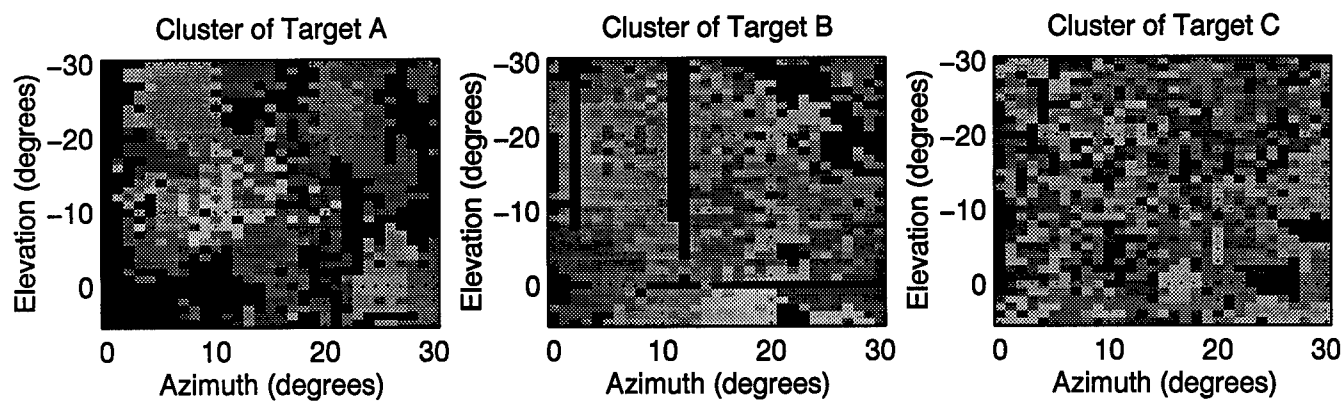
Figure 26. Clustering results for 5x5 averaging initialization for total aspect window
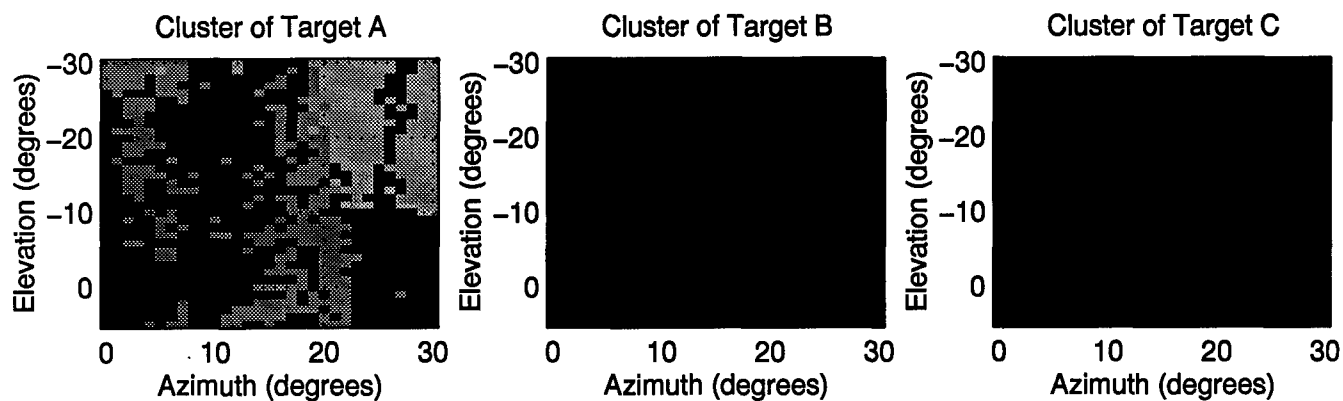
Figure 27. Clustering results for Maxi-min initialization for total aspect window
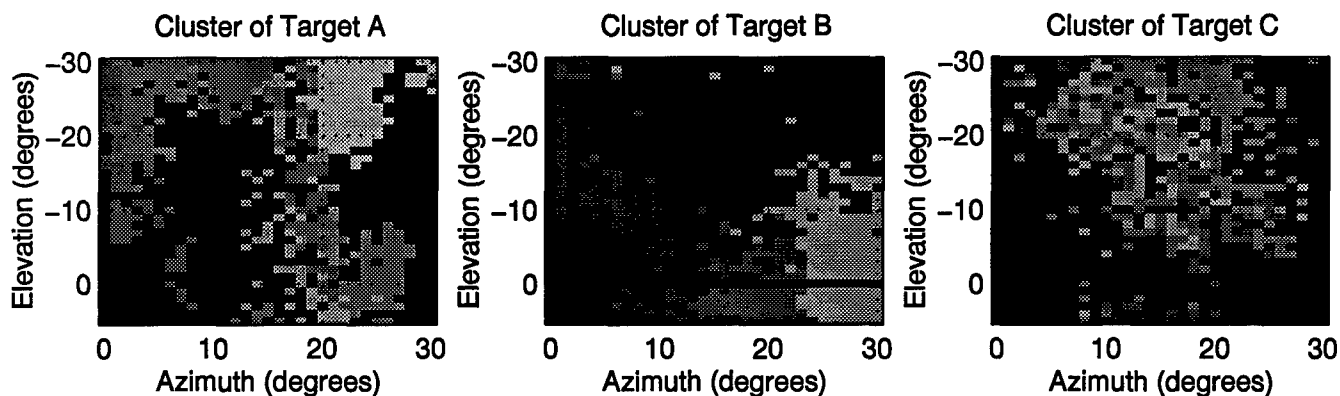
Figure 28. Clustering results for KLI initialization for total aspect window
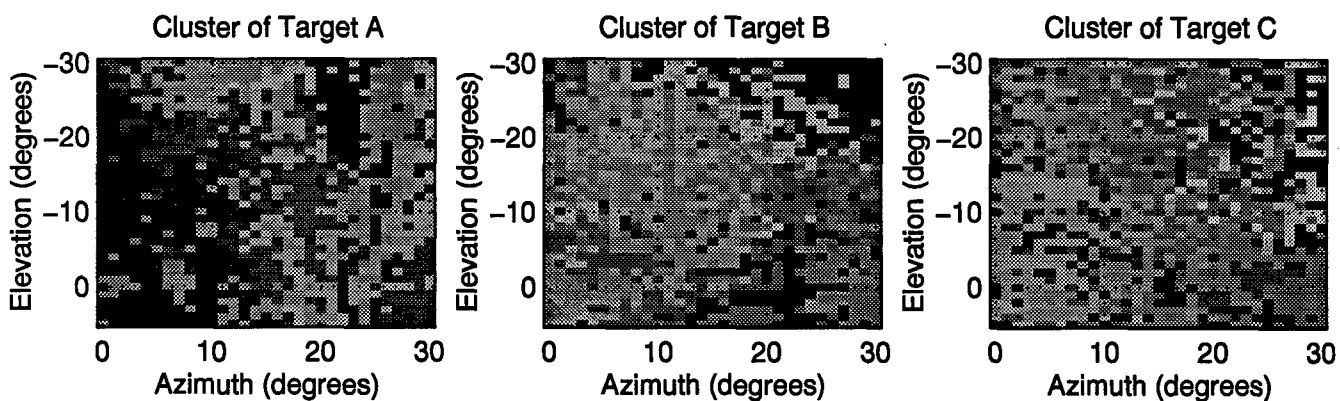
Figure 29. Clustering results for Kohonen with 6x7 output nodes for total aspect window

dows. Therefore, by averaging arbitrary groups of signatures like those done with the arbritary 5x5 windows we have a good chance of under-characterization in our database. For instance, looking at aspect window 1 to 5 degrees elevation by 0 to 5 degrees azimuth in Figures 17 - 21, we can see that this window contains aspects that belong to many different clusters. Grouping all these aspects into one template would be very undesirable and cause loss of information due to under-characterization.

Another observation is that every clustering variation has different final clusters. We noted earlier that depending on how we initialize our set of initial parameters the final results would be different. But if we examine the clustering results closely we see that their are also some basic similarities in the final results. For instance, if we look at each Figure in the 11 to 12 degrees azimuth over -10 to -30 elevation we see that these signatures always group to the same cluster. Many other similar grouping exists and is consistent between the clustering techniques.

We must always keep in mind that good representation does not always mean good classification performance. In the next section we will look at the classification results of our three class problem using a VQ classifier described in Chapter III. The classifier will use the templates created from our different clustering techniques.

5.7 Classification Performance

This section presents and summarizes the results obtain from applying the methodology and classifier described in Chapter III. In Sections 5.7.1, 5.7.2 and 5.7.3 results are presented in the form of confusion matrices for the full, three-class comparisons. To determine the variabilities of our results, a 97.5% confidence intervals are given for all estimates of the classification rates. In Section

68

5.7.4 we discussed some integration techniques that can be used to improve the classification results obtain in Sections 5.7.2 and 5.7.3. In Section 5.7.5 we discussed some of the implications of our results.

5.7.1 Results for Adaptive Gaussian Classifier.   In Section 2.3.1 we discussed the work done by Kosir and DeWall (19) and how they concluded that the use of a Quadratic discriminant function in the AGC did worse than using just a Euclidean discriminant function with the AGC. They showed that we gained little information and in fact actually gained more distortion by using the variances of the data. To confirm these results we will also run our version of the AGC with both the Quadratic discriminant function given in Equation (23) and then run the experiment over again setting the variances to one. The results of both experiments are given in Tables 1 and 2. These results confirmed the conclusion of Kosir and Dewall, that using Euclidean will give better classification results then that of a Quadratic for our data and problem.

5.7.2 Baseline results us 5x5 azimuth/elevation templates.   We will use the classification results from using the 5x5 azimuth/elevation window templates for our baseline results. This is a good baseline to see if clustering can improve on our classification results, since this is the same technique for templates creation currently being employed by Wright Laboratory. See Table 3.

5.7.3 Batch k-means and Kohonen performance results.   Taking the templates created from the k-means and Kohonen clustering techniques shown in Sections 5.4 and 5.5 and employing a VQ to our three class problem. The classification results are given in Tables 4, 5, 6, 7, and 8.

5.7.4 Integration.   To further increase classification performance we could employ a couple of different integration schemes to improve our classifier. What we could do is take a 5 out of 8
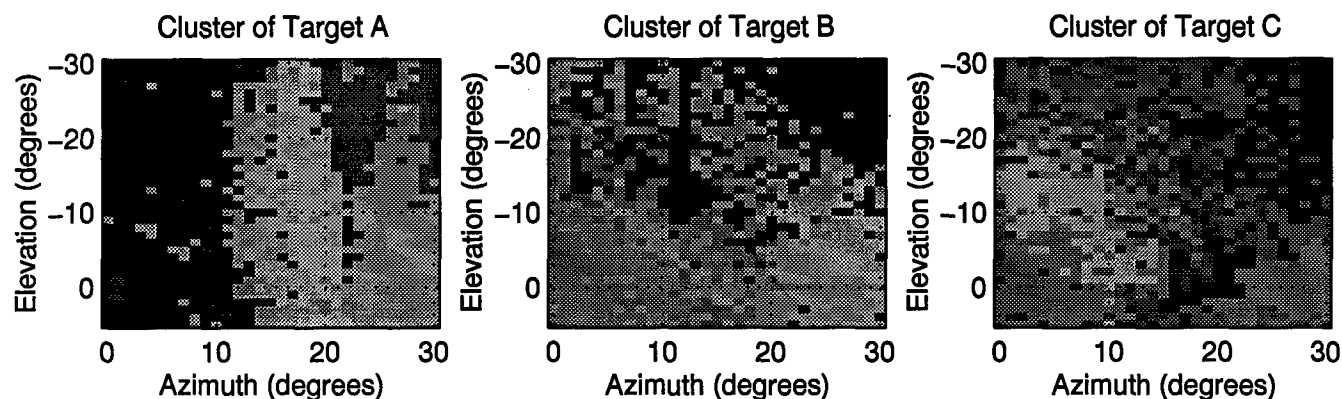
Figure 30. Clustering results for Kohonen with 1x42 output nodes for total aspect window

Actual Class	Assigned Class			P_c (%)	97.5 % Confidence Interval (%)
	Target A	Target B	Target C		
Target A	6736	164	4	97.57	± 0.36
Target B	1045	3563	107	75.57	± 1.23
Target C	1240	2413	1156	24.04	± 1.21

Table 1. Results for Adaptive Gaussian Classifier with Quadratic Discriminant Function

Actual Class	Assigned Class			P_c (%)	97.5 % Confidence Interval (%)
	Target A	Target B	Target C		
Target A	6681	111	112	96.77	± 0.42
Target B	65	3726	924	79.02	± 1.16
Target C	0	3	4806	99.94	± 0.06

Table 2. Results for Adaptive Gaussian Classifier with Euclidean Discriminant Function

Actual Class	Assigned Class			P_c (%)	97.5 % Confidence Interval (%)
	Target A	Target B	Target C		
Target A	5064	974	866	73.28	± 1.05
Target B	72	4367	276	92.62	± 0.75
Target C	10	1678	3121	64.90	± 1.35

Table 3.   Baseline results with a Vector Quantizer using 5x5 window templates and confusion table

| Actual | Assigned Class | | | P_c | 97.5 % Confidence |
Class	Target A	Target B	Target C	(%)	Interval (%)
Target A	5089	1047	768	73.71	± 1.04
Target B	25	4231	459	89.73	± 0.87
Target C	5	1379	3425	71.22	± 1.28

Table 4. Batch K-mean with 5x5 Average Initialization classification results and confusion table

| Actual | Assigned Class | | | P_c | 97.5 % Confidence |
Class	Target A	Target B	Target C	(%)	Interval (%)
Target A	6081	465	358	88.08	± 0.76
Target B	62	4519	134	95.84	± 0.57
Target C	80	1422	3307	68.77	± 1.31

Table 5. Batch K-mean with Maxi-min Initialization classification results and confusion table

| Actual | Assigned Class | | | P_c | 97.5 % Confidence |
Class	Target A	Target B	Target C	(%)	Interval (%)
Target A	5404	628	872	78.27	± 0.97
Target B	74	4253	388	90.20	± 0.85
Target C	14	1146	3649	75.88	± 1.21

Table 6. Batch K-mean with KLI Initialization classification results and confusion table

| Actual | Assigned Class | | | P_c | 97.5 % Confidence |
Class	Target A	Target B	Target C	(%)	Interval (%)
Target A	6004	307	593	86.96	± 0.79
Target B	205	4206	304	89.20	± 0.89
Target C	43	1367	3399	70.68	± 1.29

Table 7. Kohonen with 6x7 output layer nodes classification results and confusion table

| Actual | Assigned Class | | | P_c | 97.5 % Confidence |
Class	Target A	Target B	Target C	(%)	Interval (%)
Target A	6156	274	474	89.17	± 0.73
Target B	72	4437	206	94.10	± 0.67
Target C	43	1458	3308	68.79	± 1.31

Table 8. Kohonen with 1x42 output layer nodes classification results and confusion table

or a full integration of the single-look performance and build confidence in our decision. What we mean by single-look performance, is that for every input vector we get, we will make a decision on its class. For a 5 out of 8 scheme we would take 8 single-look decisions and if at least 5 out of the eight single-look decisions are of the same class than we would classify the target as that class. If there are less than 5 out of 8 than we would make no declaration. In the 5 out of 8 scheme, as we continue to get more inputs we would add this new input to our group of eight and throw away the oldest input and than make a new decision. In full integration we would not throw away the oldest input, but make a decision on the class if there are (5/8)% or 62.5% of all the decisions are of the same class.

The results are given for each templates creation techniques and both the 5 out of 8 and full integration results are shown. P_d stands for probability of declaration. Which means for every opportunity to declare, how often do we make a decision of the target class. P_c stands for probability of correct classification. Which means for every declaration, how often do we make the right decision of the target class.

See Tables 9, 10, 11, 12, 13, and 14 for the integration results.

5.7.5 Discussion of Classification Results. The objective of this section was to investigate the use of clustering techniques to enhance algorithm's performance with synthetic database range profiles and testing on measured range profiles. Our results shows that the algorithm's performance on templates created by clustering synthetic signatures was either enhanced or equivalent to the baseline in most cases. Up to a 15% increase in performance was achieved for class A and up to a 10% increase in class C depending on clustering techniques. Class B had no significant changes in performance, but this is probably attributed to the high baseline performance of class B. While

72

5x5 window templates				
Target Class	5 out of 8 results		full integration results	
	P_d (%)	P_c (%)	P_d (%)	P_c (%)
Target A	83.27	93.35	100	100
Target B	97.81	99.72	100	100
Target C	87.11	73.61	94.42	100

Table 9. 5 out of 8 and full integration Performance Results for 5x5 window templates

5x5 Average Initialization				
Target Class	5 out of 8 results		full integration results	
	P_d (%)	P_c (%)	P_d (%)	P_c (%)
Target A	83.63	93.50	100	100
Target B	97.71	99.13	100	100
Target C	89.05	82.86	100	100

Table 10. 5 out of 8 and full integration Performance Results for 5x5 Average Initialization

Maxi-min Initialization				
Target Class	5 out of 8 results		full integration results	
	P_d (%)	P_c (%)	P_d (%)	P_c (%)
Target A	95.66	99.15	99.94	100
Target B	99.70	100	100	100
Target C	86.88	79.31	100	100

Table 11. 5 out of 8 and full integration Performance Results for Maxi-min Initialization

KLI Initialization				
Target Class	5 out of 8 results		full integration results	
	P_d (%)	P_c (%)	P_d (%)	P_c (%)
Target A	87.81	97.06	100	100
Target B	97.58	99.50	100	100
Target C	91.27	87.22	100	100

Table 12. 5 out of 8 and full integration Performance Results for KLI Initialization

6x7 Kohonen				
Target Class	5 out of 8 results		full integration results	
	P_d (%)	P_c (%)	P_d (%)	P_c (%)
Target A	94.49	98.37	100	100
Target B	96.43	99.23	100	100
Target C	88.09	81.63	100	100

Table 13. 5 out of 8 and full integration Performance Results for 6x7 Kohonen

improving performance overall, there was no significant instances of detriment to performance cause by clustering.

The integration performance showed that great improvement is attainable from using the techniques. The drawback is that computation time for a decision is also increased. This could be a major drawback if only a few seconds could determine the outcome of an engagement.

5.8 Data reduction study

In this section we investigated the potential of reducing the total number of required templates per class that must be stored in the database with minimum loss of information. In this experiment we chose the Kohonen clustering algorithm as the method to generate the required number of codewords M. We chose the Kohonen algorithm with a 1XM output layer since from our performance results the Kohonen with a 1X42 achieved very good results. We chose M to be 1, 2, 3, 4, 5, 8, 10, 12, 15, 20, 25, 30, 35, and 42. We then graphed the total average distortions versus the number of codewords M. This plot is shown in Figure 31.

From Figure 31 we see that the average distortion falls as we increase the number of codewords, but at a certain point we ceased to lower the distortion. This break point is somewhere around 10 codewords for our aspect window of interests, any less than that and we have under-characterization and anymore and we have over-characterization. The classification results with only 10 codewords per class is given in Figure 15. These results are comparable to those results using 42 codewords per class.

1x42 Kohonen				
Target Class	5 out of 8 results		full integration results	
	P_d (%)	P_c (%)	P_d (%)	P_c (%)
Target A	96.38	98.83	100	100
Target B	99.17	100	100	100
Target C	87.90	79.18	100	100

Table 14. 5 out of 8 and full integration Performance Results for 1x42 Kohonen
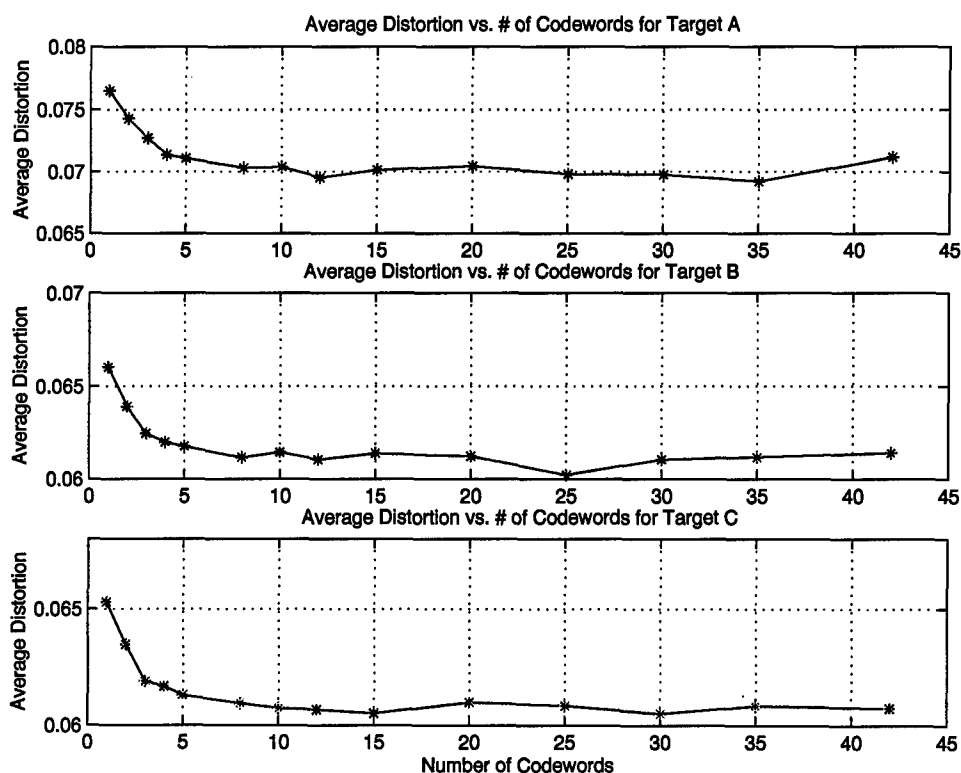
Figure 31. Average Distortion versus Total Number of Codewords

Actual Class	Assigned Class			P_c (%)	97.5 % Confidence Interval (%)
	Target A	Target B	Target C		
Target A	5797	469	638	83.97	± 0.87
Target B	60	4445	210	94.25	± 0.66
Target C	41	1712	3056	63.55	± 1.36

Table 15. Results for Kohonen with 1X10 output layer

5.9 Chapter Summary

This chapter shows all the results which accomplished the goals of this thesis: To improve classification accuracy while reducing the amount of data required to do this classification. The results showed that we can increase performance in some cases as much as $10\% - 15\%$ while being able to reduce the size of the data base by a factor of 4.

VI. Conclusion

6.1 Introduction

The primary objectives of this research was to examine the use of clustering techniques to enhance algorithm performance with synthetic database range profiles and to reduce the total number of required templates per class that must be stored in the database. Both of these objectives were met, by applying two unsupervised algorithms (k-means and Kohonen). The two clustering techniques have been validated experimentally using problems with known distributions, and the two techniques have been applied to the HRR problem. Chapter II provides the theoretical background for the two methods, which are presented in Chapter III. Chapter IV presents the results obtained by each of the methods on known toy problems. Chapter V presents the performance results for the clustering techniques using a Vector Quantizer classifier. The purpose of this chapter is to summarize the results of this research and to draw some general conclusions based on these results.

6.2 Contributions

- This thesis introduced a method for integrating clustering techniques into templates creation to improve algorithm performance and minimize database size. Instead of creating templates, basing the group of signatures to be averaged on arbitrary regions of viewing aspects, the averages are taken over the signatures contained in the natural clusters which can be identify.

- The analysis showed that the proper clustering of signatures into natural clusters can improve classification performance between 10% to 15% for some classes. It also showed that by clustering that the performance was not significantly lower in any of the cases.

77

- The analysis also showes that the proper clustering of signatures into natural clusters can reduce the database required by a factor of four. This would lead to decreased algorithm's memory and computational requirements with a consequent decrease in the time required to perform the required identification calculations.

- This thesis also applied a VQ classifier to the HRR identification problem with very good results. With the use of the VQ results we were able to explore the value of single look classification versus multi-look (integration) classification. The use of multi-look classification can prove crucial in maintaining high performance as the number of classes are increased to a usable size.

6.3 Summary of Results

Summary of Results						
Target class	Target A		Target B		Target C	
	P_c (%)	$\triangle$ (%)	P_c (%)	$\triangle$ (%)	P_c (%)	$\triangle$ (%)
Baseline Result of VQ using 5x5 window templates	73.28		92.62		64.90	
Batch k-means with 5x5 Avg. Init.	73.71	0.43	89.73	-2.88	71.22	6.32
Batch k-means with Maxi-min Init.	88.08	14.80	95.84	3.22	68.77	3.87
Batch k-means with KLI Init.	78.27	4.99	90.20	-2.42	75.88	10.98
Kohonen with 6x7 output layer nodes	86.96	13.68	89.20	-3.41	70.68	5.78
Kohonen with 1x42 output layer nodes	89.17	15.89	94.10	1.48	68.79	3.89
Kohonen with 1x10 output layer nodes	83.97	10.69	94.25	1.63	63.55	-1.35

Table 16. Summary of Results

The results of Chapter V experiments are summarized in Table 16. The performance is reported for each experiment for each target and the $\triangle$% is the value of increased or decreased in performance as compared to the baseline performance.

- Each clustering iteration did improve performance over the baseline results with a 5x5 window adhoc template formation, while not causing any significant detriment to performance.

78

- As we noted earlier that, depending on how we initialized our clustering algorithms the final clusters will be somewhat unique. This representation even though different for each clustering iteration is a better method for template creation to improve data representation and classification.

- Our experiment at data reduction, using only 10 codewords rather than 42 codewords, also was quite successful. We were able to maintain the classification performance while reducing the amount of data stored in the database by a factor of four.

- All of these classification performances can be increased by the use of a multi-look or integration classifier schemes. In fact for our three class problem we were able to attain 100% performance on all classes for all clustering schemes and for the baseline.

6.4 Follow-on Research

The research discussed in this thesis is by no means exhaustive. As with any large undertaking, there are many areas left for further research. Some possible enhancements are listed below:

- The number of classes in this investigation was limited to just three classes of fighter size aircraft. Further investigation on a more complex problem with a larger number of mixed size aircraft could be interesting. Clustering with a more complex problem could prove to have greater payoff on performance, database size, and computational complexity and time.

- The aspect window investigated was also very limited. Expansion of the aspect window to include the entire sphere of possible aspect angle should be done.

- Investigation needs to be done to see if integration classifiers can be operationally viable. Can we perform classification in a timely manner that will give the warfighter the necessary time to act on the information?

- The integration experiment showed that most of the input vectors gave correct classifications. There are a few input vectors that are spread out that are some how distorted that gave the incorrect classifications. This distortion can be caused by misalignment, antenna flashes, EMI/RFI, atmospheric interference, clutter, radar calibration, low signal to noise and etc. Further investigation needs to look at either prescreening input signatures for these problems and eliminating the distortion or the input signatures itself.

- One of the main areas of possible improvement to the HRR problem is to lower the dimensionality of the problem by doing features selection and saliency. Using every range bin is not a very viable feature in that it is so dependent on alignment and the information is only located in a small subset of the range bins.

- We know that there are differences between the synthetic generated signatures and the measured signatures. Even though the synthetic signatures do not completly model every characteristic correctly there are many characteristics that are modeled well. Therefore we need to find the common information space between the synthetic and measure signatures and use this information to come up with good features for our classifiers or use this information to help improve the synthetic predictions.

6.5 Conclusion

The results obtained in this thesis lead to several interesting conclusions:

- The use of k-means and Kohonen for clustering of the HRR data proved to significantly increase classification performances in most cases, while no significant detriment in performances was observed.

- The clustering of signatures can create an efficient database which capture the intrinsic variability of the HRR signatures without either loss of information or without excessive redundancy.

- The benefits of applying natural cluster to classification algorithms are increased performances, decreased algorithm's memory and computational requirements with a consequent decrease in the time required to perform the required identification calculations.

With these conclusions, this thesis meets each of the objectives outlined in Section 1.6. The two methods outlined here may be applied to virtually any pattern recognition problem where the grouping of "similar" objects could help with classification or reduced redundancy and the amount of data stored.

Appendix A. K-means and Kohonen Code

This appendix contains the MATLAB code which implements the k-means approaches. `turbomm.m` calls `findfar.m` to compute the initial codewords using the maxi-min initialization scheme described in Section 3.2.2.2. `kli.m` compute the intitial codewords using the KLI inititialization scheme described in Section 3.2.2.3. `lloydref.m` is a funtion used to performs the Lloyd iteration to refine an initial codebook until the change in average distortion falls below a specified level. `quantize.m` is a Vector Quantizer classifier that uses Euclidean distance that is described in Section 3.4. The scripts `kohonenscript.m` is the shell script use to implement the Kohonen SOFM using the MATLAB Neural Network Toolbox.

A.1 turbomm.m

```
%       turbo_mm.m

%       Turbo-ized version of the maxi-min algorithm.  The turbo

%       idea is from katsavounidis

%       function  C = turbomin(A,Ncdw);

%

%       where

%               C = maximin codebook

%               A = input data

%               Ncdw = number of codewords


function  C = turbo_mm(A,Ncdw);
```

```
nd = size(A,2);

origin = zeros(1,nd);


% get first codeword; gotta compute norm of all data points

[pnt,d,index]=findfar(A,origin);

C=A(index,:);

% remove first codeword from data set

Ahat = nixrow(A,index);


% find second codeword by computing distance from first codeword

% to all remaining points


d1 = clever(Ahat,C(1,:));  % find distances from c1 to all points in Ahat


[pnt,index]=max(d1);

C(2,:) = Ahat(index,:);

c = C(2,:);

% remove second codeword from data set

Ahat = nixrow(Ahat,index);

% also gotta remove the corresponding value from the distance vector

d1 = nixcol(d1,index);


for jj = 3:Ncdw
```

```
d = clever(Ahat,c);

if (jj == 3)

    D=[d1;d];

end

if (jj >= 4)

    D=[D;d];

end

% find min distance to each codeword

dmin = min(D);

% find max of the min distances

[val,index]=max(dmin);

% the next codeword is found!

c = Ahat(index,:);

C(jj,:) = c;

% get rid of that point and distance calc

Ahat = nixrow(Ahat,index);

D=nixcol(D,index);

end
```

A.2 findfar.m

```
% findfar.m

% function [pnt,d,index] = findfar(A,x);
```

```
% where A is matrix of data

%        x is point

%        pnt is the selected point from A

%        d is the distance from x to d

%        index is index of pnt

%

function [pnt,d,index] = findfar(A,x);

% compute distances


N = size(A,1);


for i = 1:N

   delta = A(i,:) - x;

   dist(i) = norm(delta);

end


[a,b] = max(dist);


pnt = A(b,:);

d = dist(b);

index = b;
```

A.3 kli.m

```
% kli.m

% use this routine to get an initial codebook of any size

% function klicbk = kli(y,Ncdw);

% where

% y =    the set of data, one fv per row

%  Ncdw = the number of codewords to generate


function klicbk = kli(y,Ncdw);


% make data set zero mean

% first find number of fv's (# rows)

ndrows = size(y,1);


meany = mean(y);

allmeany =  ones(ndrows,1)*meany;

y = y - allmeany;


% compute covariance of zero mean data, then find eigenvectors of

% covariance matrix

covy = cov(y);

[v,d] = eig(covy);
```

```
% find the total power

P = trace(d);



% get the number of dimensions

dims = size(covy,1);



% N(i) is the number of initial codewords along dimension i



N = zeros(dims,1);

for dimindx = 1:dims % for each dimension

    N(dimindx) = round( Ncdw * d(dimindx,dimindx)/P );

end

ncheck = sum(N);

if(ncheck ~= Ncdw)

  warning = 'not right number of codewords!'

  delta_ncdw = ncheck - Ncdw

  if (delta_ncdw > 0) % find dimension with most codewords and subtract

      [a,b] = max(N)

      N(b) = N(b) - delta_ncdw;

  end

  if(delta_ncdw < 0) % find dimenstion with least codewords and add

      [a,b] = min(N);

      N(b) = N(b) - delta_ncdw; % note that this will increase N(b)
```

```
    end

end

N;

newcheck=sum(N);




% now spread codewords out along eigenvectors;

% assume spread of +/- 2sigma; this keeps codwords within the data

row = 1;

for i = 1:dims

    if (N(i) ~=0 )

    sigm =  sqrt(d(i,i));

    if (rem( N(i),2))  == 0  % then even number of codewords

        oddity = 'no'; % in this dimension

        cell = 4*sigm/N(i);

        for j = 1:N(i)/2

            icbk(row,:) = v(:,i)'* (1/2)*(2*j -1)*cell;

            icbk(row+1,:) =  -1*icbk(row,:);

            row = row+2;

        end

    end


    if (rem( N(i),2)) == 1 % then odd number of codewords
```

```
      oddity = 'yes'; % in this dimension

        if (N(i) == 1)

          icbk(row,:) = sigm*0.001* randn(1,dims);
```
% should put it at mean value

% later; make it a suggestion

% in the icassp submission

% try putting the sole surviving codeword in a dimension at

% sigm times that eigenvector, the origin will take care of

% itself, (i think)

%   icbk(row,:) = sigm*v(:,i)'; doesn't seem to help

```
 row = row+1; % added this fix on 26 july

      else

 icbk(row,:) = sigm*0.001*randn(1,dims);

 row = row+1;

        cell = 4*sigm/(N(i)-1);

        for j = 1: (N(i)-1)/2

          icbk(row,:) = v(:,i)'*j*cell;

    icbk(row+1,:) = -1*icbk(row,:);

          row = row+2 ;

        end

      end

    end
```

```
    end % for N(i) not = 0

end


% now i gotta add the mean value to the codebook vectors


addmn = ones(Ncdw,1)*meany;

klicbk = icbk + addmn;
```

A.4   lloydref.m

```
function [codebook,D]=lloydref(X,C,theta);


% function [codebook,D]=lloydref(X,C,theta);

%

% This function performs Lloyd iterations to refine an

% initial codebook.  Iterations are performed until the

% change in average distortion falls below theta.

%

% X = training data

% C = initial codebook

% theta = threshold


L=size(C,1);
```

```
done=0;

d=zeros(L,1000);

D=zeros(1,1000);

codebook=C;

its=0;

while ~done,

%

% Form NN partitions:

%

its=its+1;

dist=zeros(L,size(X,1));

for i=1:L,

    diffsq=abs(X-ones(length(X),1)*codebook(i,:)).^2;

    dist(i,:)=sqrt(sum(diffsq'));

end; % for i

[nndist,nn]=min(dist);

%

% update codebook and distortion

%

for i=1:L,

  Xrows=find(nn==i);

  if length(Xrows) ==1

        codebook(i,:)=X(Xrows,:);
```

```
    else

        codebook(i,:)=mean(X(Xrows,:));

    end;  % if

    d(i,its+1)=mean(sum((X(Xrows,:)-ones(length(Xrows),1)*codebook(i,:))'.^2));

end;

D(1,its+1)=sum(d(:,its+1))/(size(X,1)*size(X,2));

if abs((D(1,its)-D(1,its+1))/D(1,its+1)) <= theta

    done=1;

end;


% Added to periodically save codebook by Pham 24 Jul 97

save HRR_codebooks.mat codebook D


disp('Another Iteration...');

end; % while


D=D(1:its+1);


A.5  quantize.m

function [Distortion,MinCodeWord]=quantize(VectorIn, CodeBookIn)

% function [Distortion,MinCodeword]=quantize(VectorIn, CodeBookIn)

%    Uses a distance metric based on the Euclidean norm.

%    Computes the distortion resulting from the codeword that is the
```

```
%    closest match of the input vector compared to the input codebook

%

%    Inputs:

%    VectorIn - a row vector of features

%    CodeBookIn - a matrix of size [number of codewords x length of VectorIn]

%

%    Outputs:

%    Distortion - mean squared error of closest codeword match to vector

%    MinCodeWord - The index of the codeword with the closest match


codewords=size(CodeBookIn,1);

distance=zeros(codewords,1);


for i=1:codewords

   diff=VectorIn-CodeBookIn(i,:);

   distance(i)=norm(diff);

end



[Distortion,MinCodeWord]=min(distance);
```

A.6   Typical "Driver" Script for Kohonen using the MATLAB Neural Net Toolbox

```
% kohonenscript.m: Shell for Kohonen to for clustering
```

```
clear all

load synSA.mat


% Take in the data and resample from 812 dimensions to 203 dimensions


x = sqrt((I.^2)+(Q.^2));

sigs = x';

[M,N]=size(sigs);

for n=1:N

resampsig(:,n) = resample(sigs(:,n),1,4);

end



% Normalize data


[M,N]=size(resampsig);

avgE = sum(resampsig);

y=1:M;

[E,y] = meshgrid(avgE,y);

normresampsig = resampsig./E;
```

```
% Kohonen clustering (self organizing feature map)


P = normresampsig;

Pmm = [min(P'); max(P')]';

Sx = 42;

Sy = 1;

M = nbdist(Sx,Sy);

S = Sx * Sy;

W= initsm(Pmm,S);


plotsm(W,M)

whos

tp = [50 5000 1];

W = trainsm(W,M,P,tp);


a = simusm(P,W,M,0)

a = full(a);

[MM,NN]=max(a);


quit
```

Bibliography

1. Andersh, Dennis J., S.W. Lee Robert Pinto C.L. Yu and Frederick L. Beckner. "Range Profile Sythesis For Noncooperative Target Identification," *CISC Conference Proceedings*, *2*:489 – 494 (1995).

2. Beckner, Frederick L., Darrell K. Ingram and Barton S. Wells. "Applications of Signature Clustering to Aircraft Identification Using High Range Resolution Radar," *CISC Conference Proceedings*, *2*:170 – 182 (1995).

3. Bishop, Christopher. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, 1995.

4. Cover, T. M. "Geometrical and Statistical Properties of Linear Inequalities With Application to Pattern Recognition," *IEEE Transactions in Electronic Computations*, *EC-14*:326 – 334 (June 1965).

5. DeSimio, Martin P., Dennis W. Ruck and Raymond E. Slyh. *Karhunen-Loeve Based Initialization for Generalized Lloyd Iteration*. Technical Report, Air Force Institute of Technology, 1996.

6. DeWall, Robert, Pete Kosir and Richard A. Mitchell. "Advanced Techniques for Training Ultra High Range Resolution Target Identification Systems," *CISC Conference Proceedings*, *2*:171 – 193 (1995).

7. DeWitt, Mark R. *High Range Resolution Radar Target Identification Using the Prony Model and Hidden Markov Models*. MS thesis, Air Force Institute of Technology, 1992.

8. Duda, Richard O. and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

9. Eaves, Jerry L. and Edward K. Reedy. *Principles of Modern Radar*. Van Nostrand Reinhold, 1987.

10. Eisenbies, Christopher Lawrence. *Classification of Ultra High Range Resolution Radar Using Decision Boundary Analysis*. MS thesis, Air Force Institute of Technology, 1994.

11. Estes, S. E. *Measurement Selection for Linear Descriminant Used In Pattern Classification*. Technical Report, San Jose Research Lab., April 1965.

12. Foley, Donald H. "Considerations of Sample and Feature Size," *IEEE Transactions on Information Theory,*, *IT-18*(5):618 – 626 (September 1972).

13. Fukunaga, Keinosuke. *Introduction to Statistical Pattern Recognition*. Academic Press, Inc., 1990.

14. Gregg, Daniel West. *Decision Boundary Analysis Feature Selection for Breast Cancer Diagnosis*. MS thesis, Air Force Institute of Technology, 1997.

15. Jain, Anil K. and Jianchang Mao. "Neural Networks and Pattern Recognition," *Computational Intelligence Imitating Life, IEEE Press* (1994).

16. Kanal, L. and B. Chandrasekaran. "On Dimensionality and Sample Size in Statistical Pattern Classification," *Proceedings: 1968 National Electronics Conference*, *2*(7) (1968).

17. Katsavounidis, Ioannis, C.-C. Jay Kuo and Zhen Zhang. "A New Initialization Technique for Generalized Lloyd Iteration," *IEEE Signal Processing Letters*, *1*(10):144 – 146 (October 1994).

18. Kohonen, Teuvo. *Self-Organization and Associative Memory* (Second Edition). Springer-Verlag, 1965.

19. Kosir, Pete and Robert Dewall, "AVT Support and Classifier Research," October 1997. Presentation at Hughes Aircraft Company.

20. Kouba, Eric T. *Recurrent Neural Networks for Radar Target Identification*. MS thesis, Air Force Institute of Technology, 1992.

21. Lee, Chulhee and David A. Landgrebe. "Feature Extraction Based on Decision Boundaries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(4):388 – 400 (April 1993).

22. Linde, Y., A. Buzo and R. M. Gray. "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communication*, *COM-28*(1):84 – 95 (January 1980).

23. Lippmann, Richard P. "An Introduction to Computing and Neural Nets," *IEEE ASSP Magazine*, *4* (1987).

24. Makhoul, J., S. Roucas and H. Gish. "Vector Quantization in Speech Coding," *Proc. IEEE*, *73*(11):1551 – 1588 (November 1985).

25. Mao, Jianchang and Anil K. Jain. "A Self-Organizing Network for Hyperellipsoidal Clustering," *IEEE Transactions on Neural Networks*, *7*(1):16 – 29 (January 1996).

26. Martin, Curtis Eli. *Non-Parametric Bayes Error Estimation For UHRR Target Identification*. MS thesis, Air Force Institute of Technology, 1993.

27. Mitchell, Richard A. and Rob DeWall. "Overview of high range resolution radar target identification," *Technical report, Wright Laboratories, Mission Applications Division* (1994).

28. Papoulis, Athanasios. *Probability, Random Variables, and Stochastic Processes* (3 Edition). McGraw-Hill, Inc., 1991.

29. Rogers, Steven K., Mathew Kabrisky Dennis W. Ruck and Gregory L. Tarr. *An Introduction to Biological and Artificial Neural Networks*. Bellingham, Washington: SPIE Optical Engineering Press, 1991.

30. Rogers, Steven K. "Introduction to Perceptrons: Advanced Topics in Neural Networks." April 1996.

31. Rogers, Steven K. and Dennis W. Ruck. "Feature Selection for Pattern Recognition Using Multilayer Perceptrons," *The Industrial Electronics Handbook* (1996). Innodata Publishing Services.

32. Skolnik, Merrill L. *Introduction to Radar Systems*. McGraw-Hill, Inc., 1980.

33. Tarr, Gregory L. *Multilayered Feedforward Neural Networks for Image Segmentation*. PhD dissertation, Air Force Institute of Technology, 1991.

34. Therrien, Charles W. *Decision Estimation and Classification*. John Wiley and Sons, 1989.

35. Venger, Rick, "Synthetic Data Clustering," July 1997. Presentation at Hughes Aircraft Company.

36. Younis, Khaled S. *Weighted Maholanobis Distance for Hyper-Ellipsoidal Clustering*. MS thesis, Air Force Institute of Technology, 1996.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis	

4. TITLE AND SUBTITLE APPLICATIONS OF UNSUPERVISED CLUSTERING ALGORITHMS TO AIRCRAFT IDENTIFICATION USING HIGH RANGE RESOLUTION RADAR	5. FUNDING NUMBERS
6. AUTHOR(S) Dzung Tri Pham	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GE/ENG/97D-22

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Mr. Roger L. Cranos WL/AAZI Bldg 620 2241 Avionics Circle Wright-Patterson AFB OH 45433-7318	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT Distibution Unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Identification of aircraft from high range resolution (HRR) radar range profiles requires a database of information capturing the variability of the individual range profiles as a function of viewing aspect. This database can be a collection of individual signatures or a collection of average signatures distributed over the region of viewing aspect of interest. An efficient database is one which captures the intrinsic variability of the HRR signatures without either excessive redundancy typical of single-signature databases, or without the loss of information common when averaging arbitrary groups of signatures. The identification of "natural" clustering of similar HRR signatures provides a means for creating efficient databases of either individual signatures, or of signature templates. Using a k-means and the Kohonen self organizing feature net, we identify the natural clustering of the HRR radar range profiles into groups of similar signatures based on the match quality metric used within a Vector Quantizer classification algorithm. This greatly reduces the redundancy in such databases while retaining classification performance. Such clusters can be useful in template-based algorithms where groups of signatures are averaged to produce a template. Instead of basing the group of signatures to be averaged on arbitrary regions of viewing aspect, the averages are taken over the signatures containing intake natural clusters which have been identified. The benefits of applying natural cluster identification to individual-signature HRR data preparation are decreased algorithm memory and computational requirements with a consequent decrease in the time required to perform the required identification calculations. When applied to template databases the benefits are improved identification performance.

14. SUBJECT TERMS Clustering, k-means, Kohonen SOFM, Combat Identification, Automatic Target Recognition, Fractricide, Pattern Recoginition, Recognition, Data Representation, Data Reduction			15. NUMBER OF PAGES 99
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements.*

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract   PR - Project
G - Grant   TA - Task
PE - Program   WU - Work Unit
     Element   Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with....; Trans. of....; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
Leave blank.
NASA - Leave blank.
NTIS -

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.